

Skelly GoGo

Generated by Doxygen 1.9.2



<b>1 Source content</b>	<b>1</b>
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List . . . . .	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy . . . . .	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List . . . . .	7
<b>5 File Index</b>	<b>9</b>
5.1 File List . . . . .	9
<b>6 Namespace Documentation</b>	<b>11</b>
6.1 Collision Namespace Reference . . . . .	11
6.1.1 Function Documentation . . . . .	11
6.1.1.1 BoundingBoxTest() . . . . .	11
6.1.1.2 CircleTest() . . . . .	12
6.1.1.3 CreateTextureAndBitmask() . . . . .	12
6.1.1.4 GetSpriteCenter() . . . . .	12
6.1.1.5 GetSpriteSize() . . . . .	12
6.1.1.6 PixelPerfectTest() . . . . .	12
6.1.2 Variable Documentation . . . . .	13
6.1.2.1 Bitmasks . . . . .	13
6.2 direction Namespace Reference . . . . .	13
6.2.1 Function Documentation . . . . .	13
6.2.1.1 GetOppositeDir() . . . . .	13
6.3 randomhelper Namespace Reference . . . . .	13
6.3.1 Function Documentation . . . . .	13
6.3.1.1 RandomFloatBetween() . . . . .	13
6.3.1.2 RandomIntBetween() . . . . .	14
6.4 spritehelper Namespace Reference . . . . .	14
6.4.1 Function Documentation . . . . .	14
6.4.1.1 CreateSpriteFrom() . . . . .	14
6.4.1.2 RotateSprite() . . . . .	14
6.4.1.3 SetOriginBottomCenter() . . . . .	14
6.4.1.4 SetScale() . . . . .	15
6.5 util Namespace Reference . . . . .	15
<b>7 Class Documentation</b>	<b>17</b>
7.1 Animation Class Reference . . . . .	17
7.1.1 Detailed Description . . . . .	17
7.1.2 Constructor & Destructor Documentation . . . . .	17

7.1.2.1 Animation() [1/2]	17
7.1.2.2 Animation() [2/2]	18
7.1.2.3 ~Animation()	18
7.1.3 Member Function Documentation	18
7.1.3.1 AnimationToSprite()	18
7.1.3.2 Update()	18
7.2 AnimationHandler Class Reference	20
7.2.1 Constructor & Destructor Documentation	20
7.2.1.1 AnimationHandler() [1/2]	20
7.2.1.2 ~AnimationHandler()	20
7.2.1.3 AnimationHandler() [2/2]	20
7.2.2 Member Function Documentation	21
7.2.2.1 getAnimation()	21
7.2.2.2 setAnimation()	21
7.3 Collision::BitmaskManager Class Reference	21
7.3.1 Constructor & Destructor Documentation	21
7.3.1.1 ~BitmaskManager()	21
7.3.2 Member Function Documentation	21
7.3.2.1 CreateMask()	22
7.3.2.2 GetMask()	22
7.3.2.3 GetPixel()	22
7.4 BossMonster Class Reference	22
7.4.1 Detailed Description	23
7.4.2 Constructor & Destructor Documentation	23
7.4.2.1 BossMonster() [1/2]	23
7.4.2.2 BossMonster() [2/2]	23
7.4.2.3 ~BossMonster()	23
7.4.3 Member Function Documentation	23
7.4.3.1 Attack()	24
7.4.3.2 initVariables()	24
7.4.3.3 Move()	24
7.5 BossRoom Class Reference	24
7.5.1 Detailed Description	25
7.5.2 Constructor & Destructor Documentation	25
7.5.2.1 BossRoom() [1/2]	25
7.5.2.2 BossRoom() [2/2]	25
7.5.2.3 ~BossRoom()	25
7.5.3 Member Function Documentation	25
7.5.3.1 Enter()	25
7.5.3.2 setTiles()	26
7.6 BossSpawner Class Reference	26
7.6.1 Constructor & Destructor Documentation	26

7.6.1.1 BossSpawner()	27
7.6.1.2 ~BossSpawner()	27
7.6.2 Member Function Documentation	27
7.6.2.1 SpawnMonster()	27
7.7 BowWeapon Class Reference	27
7.7.1 Constructor & Destructor Documentation	28
7.7.1.1 BowWeapon()	28
7.7.1.2 ~BowWeapon()	28
7.7.2 Member Function Documentation	28
7.7.2.1 Use()	28
7.8 Character Class Reference	28
7.8.1 Detailed Description	30
7.8.2 Constructor & Destructor Documentation	30
7.8.2.1 Character()	30
7.8.2.2 ~Character()	30
7.8.3 Member Function Documentation	30
7.8.3.1 Dead()	31
7.8.3.2 emptyList()	31
7.8.3.3 Equip()	31
7.8.3.4 generalUpdate()	31
7.8.3.5 GetAttackCooldownLeft()	31
7.8.3.6 GetAttackCooldownLength()	31
7.8.3.7 GetHitPoints()	32
7.8.3.8 GetMaxHP()	32
7.8.3.9 HasWeapon()	32
7.8.3.10 Heal()	32
7.8.3.11 Idle()	32
7.8.3.12 initVariables()	33
7.8.3.13 IsAlive()	33
7.8.3.14 Move()	33
7.8.3.15 MoveDown()	33
7.8.3.16 MoveLeft()	34
7.8.3.17 MoveRight()	34
7.8.3.18 MoveUp()	34
7.8.3.19 ResetAttackCooldown()	35
7.8.3.20 ResetCharacterToBeAlive()	35
7.8.3.21 RevertMove()	35
7.8.3.22 SetNormalSpeed()	35
7.8.3.23 shotProjectileList()	35
7.8.3.24 TakeDamage()	35
7.8.3.25 Update()	36
7.8.3.26 updateAttackCooldown()	36

7.8.4 Member Data Documentation	36
7.8.4.1 animationHandler_	36
7.8.4.2 attackCooldownLeft	36
7.8.4.3 attackCooldownLength_	36
7.8.4.4 CanAttack	37
7.8.4.5 characterProjectileType_	37
7.8.4.6 currentMaxHitpoints_	37
7.8.4.7 currentSpeed_	37
7.8.4.8 defaultMaxHitpoints_	37
7.8.4.9 defaultSpeed_	37
7.8.4.10 hasAnimation_	37
7.8.4.11 hitpoints_	37
7.8.4.12 invincibility_frame_	38
7.8.4.13 left_or_right_	38
7.8.4.14 startPos	38
7.8.4.15 weapon_	38
7.9 CollisionSystem Class Reference	38
7.9.1 Detailed Description	38
7.9.2 Constructor & Destructor Documentation	39
7.9.2.1 CollisionSystem()	39
7.9.3 Member Function Documentation	39
7.9.3.1 AddToCollisionList()	39
7.9.3.2 ProcessCollisionList()	39
7.9.3.3 RemoveFromCollisionList()	39
7.10 Entity Class Reference	39
7.10.1 Constructor & Destructor Documentation	40
7.10.1.1 Entity() [1/4]	40
7.10.1.2 Entity() [2/4]	41
7.10.1.3 Entity() [3/4]	41
7.10.1.4 Entity() [4/4]	41
7.10.1.5 ~Entity()	41
7.10.2 Member Function Documentation	41
7.10.2.1 GetBaseBoxAt()	42
7.10.2.2 GetOldPosition()	42
7.10.2.3 GetPos()	42
7.10.2.4 GetPosI()	42
7.10.2.5 GetSprite()	43
7.10.2.6 GetSpriteBounds()	43
7.10.2.7 GetSpriteCenter()	43
7.10.2.8 GetSpritePosition()	43
7.10.2.9 initSprite()	43
7.10.2.10 Render()	44

7.10.2.11 SetPos()	44
7.10.2.12 SetPosAndOldPos()	44
7.10.3 Member Data Documentation	44
7.10.3.1 oldPos_	44
7.10.3.2 pos_	44
7.10.3.3 sprite_	45
7.10.3.4 texture_	45
7.11 FloorTile Class Reference	45
7.11.1 Constructor & Destructor Documentation	45
7.11.1.1 FloorTile()	45
7.12 FrontWallTile Class Reference	46
7.12.1 Detailed Description	46
7.12.2 Constructor & Destructor Documentation	46
7.12.2.1 FrontWallTile()	46
7.13 Game Class Reference	46
7.13.1 Constructor & Destructor Documentation	47
7.13.1.1 Game()	47
7.13.1.2 ~Game()	47
7.13.2 Member Function Documentation	47
7.13.2.1 Events()	47
7.13.2.2 RenderGame()	47
7.13.2.3 Running()	48
7.13.2.4 UpdateGame()	48
7.14 Gamebar Class Reference	48
7.14.1 Constructor & Destructor Documentation	48
7.14.1.1 Gamebar() [1/2]	48
7.14.1.2 Gamebar() [2/2]	49
7.14.2 Member Function Documentation	49
7.14.2.1 Render()	49
7.14.2.2 RenderInventory()	49
7.14.2.3 Update()	49
7.15 GreenPotion Class Reference	50
7.15.1 Detailed Description	50
7.15.2 Constructor & Destructor Documentation	50
7.15.2.1 GreenPotion()	50
7.16 ICollidable Class Reference	50
7.16.1 Detailed Description	51
7.16.2 Member Enumeration Documentation	51
7.16.2.1 EntityType	51
7.16.3 Constructor & Destructor Documentation	51
7.16.3.1 ~ICollidable()	51
7.16.4 Member Function Documentation	51

7.16.4.1 GetBoundingBox()	52
7.16.4.2 GetEntityType()	52
7.16.4.3 ProcessCollision()	52
7.17 util::IPlatform Struct Reference	52
7.17.1 Constructor & Destructor Documentation	52
7.17.1.1 ~IPlatform()	53
7.17.2 Member Function Documentation	53
7.17.2.1 getRefreshRate()	53
7.17.2.2 getScreenScalingFactor()	53
7.17.2.3 setIcon()	53
7.17.2.4 toggleFullscreen()	53
7.18 LevelUpInstance Class Reference	54
7.18.1 Constructor & Destructor Documentation	54
7.18.1.1 LevelUpInstance()	54
7.18.2 Member Function Documentation	54
7.18.2.1 GainXP()	54
7.18.2.2 GetHPModifier()	55
7.18.2.3 GetLevel()	55
7.18.2.4 LevelUp()	55
7.19 LevelUpSystem Class Reference	55
7.19.1 Member Function Documentation	56
7.19.1.1 AddCharacter()	56
7.19.1.2 GainXP()	56
7.19.1.3 GetHPModifier()	56
7.19.1.4 GetLevel()	57
7.19.1.5 LevelUp()	57
7.19.2 Member Data Documentation	57
7.19.2.1 characterLevelMap	57
7.20 util::LinuxPlatform Struct Reference	58
7.20.1 Constructor & Destructor Documentation	58
7.20.1.1 LinuxPlatform()	58
7.20.2 Member Function Documentation	58
7.20.2.1 getRefreshRate()	58
7.20.2.2 getScreenScalingFactor()	58
7.20.2.3 setIcon()	59
7.20.2.4 toggleFullscreen()	59
7.21 Map Class Reference	59
7.21.1 Constructor & Destructor Documentation	60
7.21.1.1 Map()	60
7.21.1.2 ~Map()	60
7.21.2 Member Function Documentation	60
7.21.2.1 CreateDungeon()	60



7.21.2.2 DirToVec()	60
7.21.2.3 GetCurrentRoom()	61
7.21.2.4 GetRoomInDir()	61
7.21.2.5 GetSpawnRoom()	61
7.21.2.6 IsBossRoomCleared()	62
7.21.2.7 MovePlayer()	62
7.21.2.8 RenderCurrentRoom()	62
7.21.2.9 ResetMap()	62
7.22 Monster Class Reference	63
7.22.1 Detailed Description	64
7.22.2 Constructor & Destructor Documentation	64
7.22.2.1 ~Monster()	64
7.22.2.2 Monster() [1/2]	64
7.22.2.3 Monster() [2/2]	64
7.22.3 Member Function Documentation	64
7.22.3.1 Attack()	64
7.22.3.2 clampPosToRoom()	65
7.22.3.3 getDistanceToPlayer()	65
7.22.3.4 GetPlayer()	65
7.22.3.5 initVariables()	65
7.22.3.6 inRangeOfPlayer()	65
7.22.3.7 Move()	65
7.22.3.8 moveTowardsPlayer()	66
7.22.3.9 Render()	66
7.22.3.10 ReturnPotion()	66
7.22.3.11 SetTarget()	66
7.22.3.12 Update()	66
7.22.4 Member Data Documentation	66
7.22.4.1 healthbar_	67
7.22.4.2 movedLastTick_	67
7.22.4.3 player_	67
7.22.4.4 staticDamage_	67
7.23 MonsterSpawner Class Reference	67
7.23.1 Constructor & Destructor Documentation	68
7.23.1.1 MonsterSpawner() [1/2]	68
7.23.1.2 MonsterSpawner() [2/2]	68
7.23.1.3 ~MonsterSpawner()	68
7.23.2 Member Function Documentation	68
7.23.2.1 GetMonsterAmount()	68
7.23.2.2 getRandomMonster()	68
7.23.2.3 SetMonsterAmount()	69
7.23.2.4 SpawnMonster()	69

7.23.3 Member Data Documentation	69
7.23.3.1 monsterCount_	69
7.23.3.2 monsterTypeCount_	69
7.24 Collision::OrientedBoundingBox Class Reference	69
7.24.1 Constructor & Destructor Documentation	70
7.24.1.1 OrientedBoundingBox()	70
7.24.2 Member Function Documentation	70
7.24.2.1 ProjectOntoAxis()	70
7.24.3 Member Data Documentation	70
7.24.3.1 Points	70
7.25 Player Class Reference	70
7.25.1 Detailed Description	71
7.25.2 Constructor & Destructor Documentation	71
7.25.2.1 Player()	71
7.25.2.2 ~Player()	71
7.25.3 Member Function Documentation	72
7.25.3.1 AddPotion()	72
7.25.3.2 Attack()	73
7.25.3.3 ClearInventory()	73
7.25.3.4 Dash()	73
7.25.3.5 GetDashCooldownLeft()	73
7.25.3.6 GetDashCooldownLength()	73
7.25.3.7 GetInventory()	74
7.25.3.8 initVariables()	74
7.25.3.9 ResetDashCooldown()	74
7.25.3.10 Update()	74
7.25.3.11 UsePotion()	74
7.25.4 Member Data Documentation	75
7.25.4.1 CanDash	75
7.25.4.2 IsDashing	75
7.26 Potion Class Reference	75
7.26.1 Constructor & Destructor Documentation	76
7.26.1.1 Potion()	76
7.26.1.2 ~Potion()	76
7.26.2 Member Function Documentation	76
7.26.2.1 GetColour()	76
7.26.2.2 GetHealthIncrease()	76
7.26.3 Member Data Documentation	77
7.26.3.1 colour_	77
7.26.3.2 healthIncrease_	77
7.27 PowerUp Class Reference	77
7.27.1 Constructor & Destructor Documentation	77

7.27.1.1 PowerUp()	77
7.27.1.2 ~PowerUp()	77
7.28 Projectile Class Reference	78
7.28.1 Member Enumeration Documentation	79
7.28.1.1 Type	79
7.28.2 Constructor & Destructor Documentation	79
7.28.2.1 Projectile()	79
7.28.3 Member Function Documentation	79
7.28.3.1 GetDamage()	80
7.28.3.2 GetDirection()	80
7.28.3.3 GetDistanceLifeSpan()	80
7.28.3.4 GetProjectileSpeed()	80
7.28.3.5 GetStartPosition()	81
7.28.3.6 GetTimeExisted()	81
7.28.3.7 GetTimeLifeSpan()	81
7.28.3.8 GetType()	81
7.28.3.9 HasHit()	81
7.28.3.10 Hit()	82
7.28.3.11 IsAlive()	82
7.28.3.12 Kill()	82
7.28.3.13 Penetrates()	83
7.28.3.14 SetDamage()	83
7.28.3.15 SetDirection()	83
7.28.3.16 SetDistanceLifeSpan()	83
7.28.3.17 SetProjectileSpeed()	84
7.28.3.18 SetSprite()	84
7.28.3.19 SetTimeLifeSpan()	84
7.28.3.20 SetType()	85
7.28.3.21 Update()	85
7.29 RandomMonster Class Reference	85
7.29.1 Detailed Description	86
7.29.2 Constructor & Destructor Documentation	86
7.29.2.1 RandomMonster() [1/2]	86
7.29.2.2 RandomMonster() [2/2]	86
7.29.2.3 ~RandomMonster()	86
7.29.3 Member Function Documentation	86
7.29.3.1 Attack()	87
7.29.3.2 initVariables()	87
7.29.3.3 Move()	87
7.30 RedPotion Class Reference	87
7.30.1 Detailed Description	88
7.30.2 Constructor & Destructor Documentation	88

7.30.2.1 RedPotion()	88
7.31 RoomInstance Class Reference	88
7.31.1 Detailed Description	90
7.31.2 Constructor & Destructor Documentation	90
7.31.2.1 RoomInstance() [1/3]	90
7.31.2.2 RoomInstance() [2/3]	90
7.31.2.3 RoomInstance() [3/3]	90
7.31.2.4 ~RoomInstance()	90
7.31.3 Member Function Documentation	90
7.31.3.1 AddPotion()	91
7.31.3.2 Connect()	91
7.31.3.3 CreateExit()	91
7.31.3.4 deleteMonster()	91
7.31.3.5 Enter()	92
7.31.3.6 Exit()	92
7.31.3.7 GetCoords()	92
7.31.3.8 GetEntranceInDirection()	92
7.31.3.9 GetMonsters()	92
7.31.3.10 GetPotions()	93
7.31.3.11 GetRoomInDir()	93
7.31.3.12 getRoomTilesAt()	93
7.31.3.13 getTiles()	93
7.31.3.14 HasDirectionsLeft()	93
7.31.3.15 IsCleared()	94
7.31.3.16 IsVisisted()	94
7.31.3.17 monsterCleared()	94
7.31.3.18 positionIsPenetratable()	94
7.31.3.19 positionIsWalkable()	94
7.31.3.20 RemoveDirection()	94
7.31.3.21 RemoveRandomDirection()	95
7.31.3.22 Render()	95
7.31.3.23 renderSpriteBackground()	95
7.31.3.24 setTiles()	95
7.31.4 Member Data Documentation	96
7.31.4.1 cleared_	96
7.31.4.2 coords_	96
7.31.4.3 directionsLeft_	96
7.31.4.4 monsters_	96
7.31.4.5 potions_	96
7.31.4.6 roomBackground	96
7.31.4.7 roomSize_	96
7.31.4.8 roomTexture	97

7.31.4.9 spawner_ . . . . .	97
7.31.4.10 tileVector_ . . . . .	97
7.31.4.11 visited_ . . . . .	97
7.32 RoomTile Class Reference . . . . .	97
7.32.1 Constructor & Destructor Documentation . . . . .	98
7.32.1.1 RoomTile() . . . . .	98
7.32.2 Member Function Documentation . . . . .	98
7.32.2.1 getBoundingBox() . . . . .	98
7.32.2.2 getPosition() . . . . .	98
7.32.2.3 getSize() . . . . .	98
7.32.2.4 getSprite() . . . . .	99
7.32.2.5 isPenetrable() . . . . .	99
7.32.2.6 isWalkable() . . . . .	99
7.33 ScreenText Class Reference . . . . .	99
7.33.1 Constructor & Destructor Documentation . . . . .	99
7.33.1.1 ScreenText() . . . . .	100
7.33.1.2 ~ScreenText() . . . . .	100
7.34 SearchingMonster Class Reference . . . . .	100
7.34.1 Detailed Description . . . . .	101
7.34.2 Constructor & Destructor Documentation . . . . .	101
7.34.2.1 SearchingMonster() [1/2] . . . . .	101
7.34.2.2 SearchingMonster() [2/2] . . . . .	101
7.34.2.3 ~SearchingMonster() . . . . .	101
7.34.3 Member Function Documentation . . . . .	101
7.34.3.1 Attack() . . . . .	101
7.34.3.2 initVariables() . . . . .	102
7.34.3.3 Move() . . . . .	102
7.35 SlowMonster Class Reference . . . . .	102
7.35.1 Detailed Description . . . . .	103
7.35.2 Constructor & Destructor Documentation . . . . .	103
7.35.2.1 SlowMonster() [1/2] . . . . .	103
7.35.2.2 SlowMonster() [2/2] . . . . .	103
7.35.2.3 ~SlowMonster() . . . . .	103
7.35.3 Member Function Documentation . . . . .	103
7.35.3.1 Attack() . . . . .	103
7.35.3.2 initVariables() . . . . .	104
7.35.3.3 Move() . . . . .	104
7.36 SnipingMonster Class Reference . . . . .	104
7.36.1 Detailed Description . . . . .	104
7.36.2 Constructor & Destructor Documentation . . . . .	105
7.36.2.1 SnipingMonster() [1/2] . . . . .	105
7.36.2.2 SnipingMonster() [2/2] . . . . .	105

7.36.2.3 ~SnipingMonster()	105
7.36.3 Member Function Documentation	105
7.36.3.1 Attack()	105
7.36.3.2 initVariables()	105
7.36.3.3 Move()	106
7.37 SoundEffect Class Reference	106
7.37.1 Constructor & Destructor Documentation	106
7.37.1.1 SoundEffect()	106
7.37.1.2 ~SoundEffect()	106
7.37.2 Member Function Documentation	106
7.37.2.1 PlaySound()	107
7.38 StartingRoom Class Reference	107
7.38.1 Constructor & Destructor Documentation	107
7.38.1.1 StartingRoom() [1/2]	107
7.38.1.2 StartingRoom() [2/2]	108
7.38.1.3 ~StartingRoom()	108
7.38.2 Member Function Documentation	108
7.38.2.1 setTiles()	108
7.39 SwordWeapon Class Reference	108
7.39.1 Constructor & Destructor Documentation	109
7.39.1.1 SwordWeapon()	109
7.39.1.2 ~SwordWeapon()	109
7.39.2 Member Function Documentation	109
7.39.2.1 Use()	109
7.40 TreasureRoom Class Reference	109
7.40.1 Detailed Description	110
7.40.2 Constructor & Destructor Documentation	110
7.40.2.1 TreasureRoom() [1/2]	110
7.40.2.2 TreasureRoom() [2/2]	110
7.40.2.3 ~TreasureRoom()	110
7.40.3 Member Function Documentation	110
7.40.3.1 setTiles()	110
7.41 VioletPotion Class Reference	111
7.41.1 Detailed Description	111
7.41.2 Constructor & Destructor Documentation	111
7.41.2.1 VioletPotion()	111
7.42 WallPatrolMonster Class Reference	112
7.42.1 Detailed Description	112
7.42.2 Constructor & Destructor Documentation	112
7.42.2.1 WallPatrolMonster() [1/2]	112
7.42.2.2 WallPatrolMonster() [2/2]	113
7.42.2.3 ~WallPatrolMonster()	113

7.42.3 Member Function Documentation	113
7.42.3.1 Attack()	113
7.42.3.2 initVariables()	113
7.42.3.3 Move()	113
7.43 WallTile Class Reference	114
7.43.1 Constructor & Destructor Documentation	114
7.43.1.1 WallTile()	114
7.44 Weapon Class Reference	114
7.44.1 Constructor & Destructor Documentation	115
7.44.1.1 Weapon()	115
7.44.1.2 ~Weapon()	116
7.44.2 Member Function Documentation	116
7.44.2.1 AddPowerUp()	116
7.44.2.2 BoostDamageValue()	116
7.44.2.3 GetAttackCooldown()	116
7.44.2.4 getPowerUpCount()	116
7.44.2.5 GetRange()	116
7.44.2.6 SetTextureRect()	116
7.44.2.7 UnBoostDamageValue()	117
7.44.2.8 Use()	117
7.44.3 Member Data Documentation	117
7.44.3.1 attackCooldownLeft	117
7.44.3.2 attackCooldownLength_	117
7.44.3.3 cooldown_	117
7.44.3.4 currentDamage_	117
7.44.3.5 damageBoostModifier	117
7.44.3.6 defaultDamage_	118
7.44.3.7 maxPowerUps	118
7.44.3.8 penetrates_	118
7.44.3.9 powerUps_	118
7.44.3.10 projectileSize_	118
7.44.3.11 projectileSpeed_	118
7.44.3.12 range_	118
7.44.3.13 speed_	118
7.44.3.14 sprite_	119
7.44.3.15 texture_	119
7.45 util::WindowsPlatform Struct Reference	119
7.45.1 Constructor & Destructor Documentation	119
7.45.1.1 WindowsPlatform()	119
7.45.2 Member Function Documentation	120
7.45.2.1 getRefreshRate()	120
7.45.2.2 getScreenScalingFactor()	120

7.45.2.3 setIcon()	120
7.45.2.4 toggleFullscreen()	120
7.46 YellowPotion Class Reference	121
7.46.1 Detailed Description	121
7.46.2 Constructor & Destructor Documentation	121
7.46.2.1 YellowPotion()	121
<b>8 File Documentation</b>	<b>123</b>
8.1 src/Actors/character.cpp File Reference	123
8.1.1 Macro Definition Documentation	123
8.1.1.1 C_PIXELS_delta	123
8.1.1.2 C_PIXELS_X	123
8.1.1.3 C_PIXELS_Y	124
8.1.1.4 C_SCALE	124
8.1.1.5 C_X	124
8.2 src/Actors/character.hpp File Reference	124
8.2.1 Macro Definition Documentation	124
8.2.1.1 _CHARACTER_CLASS_	124
8.3 character.hpp	125
8.4 src/Actors/Monsters/BossMonster.cpp File Reference	126
8.5 src/Actors/Monsters/BossMonster.hpp File Reference	126
8.5.1 Macro Definition Documentation	126
8.5.1.1 _BOSS_MONSTER_CLASS_	126
8.6 BossMonster.hpp	126
8.7 src/Actors/Monsters/monster.cpp File Reference	127
8.8 src/Actors/Monsters/monster.hpp File Reference	127
8.8.1 Macro Definition Documentation	127
8.8.1.1 _MONSTER_CLASS_	127
8.9 monster.hpp	128
8.10 src/Actors/Monsters/MonsterSpawner/BossSpawner.cpp File Reference	128
8.11 src/Actors/Monsters/MonsterSpawner/BossSpawner.hpp File Reference	128
8.11.1 Macro Definition Documentation	129
8.11.1.1 _BOSS_SPAWNER_CLASS_	129
8.12 BossSpawner.hpp	129
8.13 src/Actors/Monsters/MonsterSpawner/MonsterSpawner.cpp File Reference	129
8.14 src/Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp File Reference	129
8.14.1 Macro Definition Documentation	130
8.14.1.1 _MONSTERSPAWNER_CLASS_	130
8.14.2 Typedef Documentation	130
8.14.2.1 MonsterSP	130
8.14.2.2 MonsterSpawnerUP	130
8.15 MonsterSpawner.hpp	131



8.16 src/Actors/Monsters/RandomMonster.cpp File Reference . . . . .	131
8.17 src/Actors/Monsters/RandomMonster.hpp File Reference . . . . .	131
8.17.1 Macro Definition Documentation . . . . .	132
8.17.1.1 _RANDOM_MONSTER_CLASS_ . . . . .	132
8.18 RandomMonster.hpp . . . . .	132
8.19 src/Actors/Monsters/SearchingMonster.cpp File Reference . . . . .	132
8.20 src/Actors/Monsters/SearchingMonster.hpp File Reference . . . . .	132
8.20.1 Macro Definition Documentation . . . . .	133
8.20.1.1 _SEARCHING_MONSTER_CLASS_ . . . . .	133
8.21 SearchingMonster.hpp . . . . .	133
8.22 src/Actors/Monsters/SlowMonster.cpp File Reference . . . . .	133
8.23 src/Actors/Monsters/SlowMonster.hpp File Reference . . . . .	133
8.23.1 Macro Definition Documentation . . . . .	134
8.23.1.1 _SLOW_MONSTER_CLASS_ . . . . .	134
8.24 SlowMonster.hpp . . . . .	134
8.25 src/Actors/Monsters/SnipingMonster.cpp File Reference . . . . .	134
8.26 src/Actors/Monsters/SnipingMonster.hpp File Reference . . . . .	135
8.26.1 Macro Definition Documentation . . . . .	135
8.26.1.1 _SNIPING_MONSTER_CLASS_ . . . . .	135
8.27 SnipingMonster.hpp . . . . .	135
8.28 src/Actors/Monsters/WallPatrolMonster.cpp File Reference . . . . .	136
8.29 src/Actors/Monsters/WallPatrolMonster.hpp File Reference . . . . .	136
8.29.1 Macro Definition Documentation . . . . .	136
8.29.1.1 _WALL_PATROL_MONSTER_CLASS_ . . . . .	136
8.30 WallPatrolMonster.hpp . . . . .	136
8.31 src/Actors/player.cpp File Reference . . . . .	137
8.32 src/Actors/player.hpp File Reference . . . . .	137
8.32.1 Macro Definition Documentation . . . . .	137
8.32.1.1 _PLAYER_CLASS_ . . . . .	137
8.32.2 Typedef Documentation . . . . .	137
8.32.2.1 PlayerPS . . . . .	137
8.33 player.hpp . . . . .	138
8.34 src/Animation/animation.cpp File Reference . . . . .	138
8.35 src/Animation/animation.hpp File Reference . . . . .	138
8.35.1 Macro Definition Documentation . . . . .	139
8.35.1.1 _ANIMATION_ . . . . .	139
8.36 animation.hpp . . . . .	139
8.37 src/Animation/Animationhandler.cpp File Reference . . . . .	139
8.38 src/Animation/Animationhandler.hpp File Reference . . . . .	139
8.38.1 Macro Definition Documentation . . . . .	140
8.38.1.1 _ANIMATIONHANDLER_CLASS_ . . . . .	140
8.38.2 Typedef Documentation . . . . .	140

8.38.2.1 AnimationPS . . . . .	140
8.38.3 Enumeration Type Documentation . . . . .	140
8.38.3.1 AnimationIndex . . . . .	140
8.39 Animationhandler.hpp . . . . .	141
8.40 src/Combat/CircularProjectile.hpp File Reference . . . . .	141
8.41 CircularProjectile.hpp . . . . .	141
8.42 src/Combat/Health/HealthPotions.hpp File Reference . . . . .	141
8.42.1 Macro Definition Documentation . . . . .	142
8.42.1.1 _HEALTHPOTIONS_CLASS_ . . . . .	142
8.43 HealthPotions.hpp . . . . .	142
8.44 src/Combat/Health/Potion.cpp File Reference . . . . .	142
8.45 src/Combat/Health/Potion.hpp File Reference . . . . .	143
8.45.1 Macro Definition Documentation . . . . .	143
8.45.1.1 _POTION_CLASS_ . . . . .	143
8.46 Potion.hpp . . . . .	143
8.47 src/Combat/PowerUps/PowerUp.hpp File Reference . . . . .	143
8.47.1 Macro Definition Documentation . . . . .	144
8.47.1.1 _POWERUP_CLASS_ . . . . .	144
8.48 PowerUp.hpp . . . . .	144
8.49 src/Combat/projectile.cpp File Reference . . . . .	144
8.50 src/Combat/Projectile.hpp File Reference . . . . .	144
8.50.1 Macro Definition Documentation . . . . .	145
8.50.1.1 _Projectile_CLASS_ . . . . .	145
8.50.2 Typedef Documentation . . . . .	145
8.50.2.1 ProjectileUP . . . . .	145
8.51 Projectile.hpp . . . . .	145
8.52 src/Combat/Weapons/BowWeapon.cpp File Reference . . . . .	146
8.53 src/Combat/Weapons/BowWeapon.hpp File Reference . . . . .	146
8.53.1 Macro Definition Documentation . . . . .	146
8.53.1.1 _BOWWEAPON_CLASS_ . . . . .	147
8.54 BowWeapon.hpp . . . . .	147
8.55 src/Combat/Weapons/SwordWeapon.cpp File Reference . . . . .	147
8.56 src/Combat/Weapons/SwordWeapon.hpp File Reference . . . . .	147
8.56.1 Macro Definition Documentation . . . . .	147
8.56.1.1 _SWORDWEAPON_CLASS_ . . . . .	147
8.57 SwordWeapon.hpp . . . . .	148
8.58 src/Combat/Weapons/Weapon.cpp File Reference . . . . .	148
8.59 src/Combat/Weapons/Weapon.hpp File Reference . . . . .	148
8.59.1 Macro Definition Documentation . . . . .	148
8.59.1.1 _WEAPON_CLASS_ . . . . .	148
8.60 Weapon.hpp . . . . .	149
8.61 src/Dungeon/map.cpp File Reference . . . . .	149

8.62 src/Dungeon/map.hpp File Reference . . . . .	149
8.62.1 Macro Definition Documentation . . . . .	150
8.62.1.1 <code>_MAP_</code> . . . . .	150
8.63 map.hpp . . . . .	150
8.64 src/Dungeon/roomInstance.cpp File Reference . . . . .	151
8.65 src/Dungeon/roomInstance.hpp File Reference . . . . .	151
8.65.1 Enumeration Type Documentation . . . . .	151
8.65.1.1 Direction . . . . .	151
8.66 roomInstance.hpp . . . . .	152
8.67 src/Dungeon/specialrooms/BossRoom.cpp File Reference . . . . .	153
8.68 src/Dungeon/specialrooms/BossRoom.hpp File Reference . . . . .	153
8.69 BossRoom.hpp . . . . .	153
8.70 src/Dungeon/specialrooms/startingRoom.cpp File Reference . . . . .	154
8.71 src/Dungeon/specialrooms/startingRoom.hpp File Reference . . . . .	154
8.71.1 Macro Definition Documentation . . . . .	154
8.71.1.1 <code>_STARTING_ROOM_CLASS_</code> . . . . .	154
8.72 startingRoom.hpp . . . . .	154
8.73 src/Dungeon/specialrooms/TreasureRoom.cpp File Reference . . . . .	154
8.74 src/Dungeon/specialrooms/TreasureRoom.hpp File Reference . . . . .	155
8.75 TreasureRoom.hpp . . . . .	155
8.76 src/Dungeon/Tiles/roomTile.cpp File Reference . . . . .	155
8.77 src/Dungeon/Tiles/roomTile.hpp File Reference . . . . .	155
8.78 roomTile.hpp . . . . .	156
8.79 src/entity.cpp File Reference . . . . .	156
8.80 src/entity.hpp File Reference . . . . .	156
8.80.1 Macro Definition Documentation . . . . .	157
8.80.1.1 <code>_ENTITY_CLASS_</code> . . . . .	157
8.81 entity.hpp . . . . .	157
8.82 src/game.cpp File Reference . . . . .	157
8.82.1 Macro Definition Documentation . . . . .	158
8.82.1.1 <code>C_PIXELS</code> . . . . .	158
8.83 src/game.hpp File Reference . . . . .	158
8.83.1 Macro Definition Documentation . . . . .	158
8.83.1.1 <code>_GAME_CLASS_</code> . . . . .	158
8.84 game.hpp . . . . .	159
8.85 src/gamebar.cpp File Reference . . . . .	159
8.86 src/gamebar.hpp File Reference . . . . .	160
8.86.1 Macro Definition Documentation . . . . .	160
8.86.1.1 <code>_GAMEBAR_CLASS_</code> . . . . .	160
8.87 gamebar.hpp . . . . .	160
8.88 src/Interfaces/CollisionSystem.hpp File Reference . . . . .	161
8.88.1 Macro Definition Documentation . . . . .	161

8.88.1.1 <code>_COLLISIONSYSTEM_CLASS_</code>	161
8.89 <code>CollisionSystem.hpp</code>	161
8.90 <code>src/Interfaces/ICollidable.hpp</code> File Reference	162
8.90.1 Macro Definition Documentation	162
8.90.1.1 <code>_COLLIDABLE_INTERFACE_</code>	162
8.91 <code>ICollidable.hpp</code>	162
8.92 <code>src/PCH.hpp</code> File Reference	163
8.92.1 Macro Definition Documentation	163
8.92.1.1 <code>NDEBUG</code>	163
8.92.1.2 <code>UNUSED</code>	163
8.93 <code>PCH.hpp</code>	164
8.94 <code>src/Platform/IPlatform.hpp</code> File Reference	164
8.95 <code>IPlatform.hpp</code>	165
8.96 <code>src/Platform/Platform.hpp</code> File Reference	165
8.97 <code>Platform.hpp</code>	165
8.98 <code>src/Platform/Unix/LinuxPlatform.cpp</code> File Reference	165
8.99 <code>src/Platform/Unix/LinuxPlatform.hpp</code> File Reference	165
8.100 <code>LinuxPlatform.hpp</code>	166
8.101 <code>src/Platform/Win32/Resource.h</code> File Reference	166
8.101.1 Macro Definition Documentation	166
8.101.1.1 <code>MANIFEST_RESOURCE_ID</code>	166
8.101.1.2 <code>WIN32_ICON_MAIN</code>	167
8.102 <code>Resource.h</code>	167
8.103 <code>src/Platform/Win32/WindowsPlatform.cpp</code> File Reference	167
8.104 <code>src/Platform/Win32/WindowsPlatform.hpp</code> File Reference	167
8.105 <code>WindowsPlatform.hpp</code>	167
8.106 <code>src/readme.md</code> File Reference	168
8.107 <code>src/Utility/Collision.cpp</code> File Reference	168
8.108 <code>src/Utility/Collision.hpp</code> File Reference	168
8.109 <code>Collision.hpp</code>	169
8.110 <code>src/Utility/FileSystem.hpp</code> File Reference	169
8.111 <code>FileSystem.hpp</code>	169
8.112 <code>src/Utility/LevelUpInstance.cpp</code> File Reference	170
8.113 <code>src/Utility/LevelUpInstance.hpp</code> File Reference	170
8.113.1 Macro Definition Documentation	170
8.113.1.1 <code>_LevelUpInstance_CLASS_</code>	170
8.114 <code>LevelUpInstance.hpp</code>	171
8.115 <code>src/Utility/LevelUpSystem.cpp</code> File Reference	171
8.116 <code>src/Utility/LevelUpSystem.hpp</code> File Reference	171
8.116.1 Macro Definition Documentation	171
8.116.1.1 <code>_LevelUpSystem_CLASS_</code>	171
8.117 <code>LevelUpSystem.hpp</code>	172

8.118 src/Utility/Main.cpp File Reference . . . . .	172
8.118.1 Function Documentation . . . . .	172
8.118.1.1 main() . . . . .	172
8.119 src/Utility/RandomHelper.cpp File Reference . . . . .	172
8.120 src/Utility/RandomHelper.hpp File Reference . . . . .	173
8.120.1 Macro Definition Documentation . . . . .	173
8.120.1.1 _RANDOMHELPER_ . . . . .	173
8.121 RandomHelper.hpp . . . . .	173
8.122 src/Utility/ScreenText.cpp File Reference . . . . .	173
8.123 src/Utility/ScreenText.hpp File Reference . . . . .	174
8.124 ScreenText.hpp . . . . .	174
8.125 src/Utility/Sounds/SoundEffects.cpp File Reference . . . . .	174
8.126 src/Utility/Sounds/SoundEffects.hpp File Reference . . . . .	174
8.127 SoundEffects.hpp . . . . .	174
8.128 src/Utility/SpriteHelper.cpp File Reference . . . . .	175
8.128.1 Macro Definition Documentation . . . . .	175
8.128.1.1 PI . . . . .	175
8.129 src/Utility/SpriteHelper.hpp File Reference . . . . .	175
8.129.1 Macro Definition Documentation . . . . .	176
8.129.1.1 _SPRITEHELPER_ . . . . .	176
8.130 SpriteHelper.hpp . . . . .	176
8.131 src/Utility/Types.hpp File Reference . . . . .	176
8.131.1 Typedef Documentation . . . . .	176
8.131.1.1 llong . . . . .	176
8.131.1.2 uchar . . . . .	177
8.131.1.3 uint . . . . .	177
8.131.1.4 ullong . . . . .	177
8.131.1.5 ushort . . . . .	177
8.132 Types.hpp . . . . .	177



## Chapter 1

### Source content

This folder should contain only `hpp/cpp` files of your implementation. You can also place `hpp` files in a separate directory `include`.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">Collision</a>	11
<a href="#">direction</a>	13
<a href="#">randomhelper</a>	13
<a href="#">spritehelper</a>	14
<a href="#">util</a>	15



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Animation	17
AnimationHandler	20
Collision::BitmaskManager	21
CollisionSystem	38
Entity	39
Character	28
Monster	63
BossMonster	22
RandomMonster	85
SearchingMonster	100
SlowMonster	102
SnipingMonster	104
WallPatrolMonster	112
Player	70
Potion	75
GreenPotion	50
RedPotion	87
VioletPotion	111
YellowPotion	121
Projectile	78
ScreenText	99
Game	46
Gamebar	48
ICollidable	50
util::IPlatform	52
util::LinuxPlatform	58
util::WindowsPlatform	119
LevelUpInstance	54
LevelUpSystem	55
Map	59
MonsterSpawner	67
BossSpawner	26
Collision::OrientedBoundingBox	69
PowerUp	77

RoomInstance . . . . .	88
BossRoom . . . . .	24
StartingRoom . . . . .	107
TreasureRoom . . . . .	109
RoomTile . . . . .	97
FloorTile . . . . .	45
FrontWallTile . . . . .	46
WallTile . . . . .	114
SoundEffect . . . . .	106
Weapon . . . . .	114
BowWeapon . . . . .	27
SwordWeapon . . . . .	108

## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Animation</a>	
<a href="#">Animation</a> build the characters animation frames from spritesheet and displays them to achive an animated sprite	17
<a href="#">AnimationHandler</a>	20
<a href="#">Collision::BitmaskManager</a>	21
<a href="#">BossMonster</a>	
This is the boss monster. It is the most powerful monster and once the player kills it the game is won	22
<a href="#">BossRoom</a>	
The roominstance where the boss spawns and which has to be cleared to beat the game	24
<a href="#">BossSpawner</a>	26
<a href="#">BowWeapon</a>	27
<a href="#">Character</a>	
<a href="#">Character</a> class that our player and monster inherits	28
<a href="#">CollisionSystem</a>	
Unused, was not intergrated or completed	38
<a href="#">Entity</a>	39
<a href="#">FloorTile</a>	45
<a href="#">FrontWallTile</a>	
A wall that is not walkable but penetrable	46
<a href="#">Game</a>	46
<a href="#">Gamebar</a>	48
<a href="#">GreenPotion</a>	
A green potion. Takes a position as parameter and the colour and healing effect is set automatically	50
<a href="#">ICollidable</a>	
Unused interface for collisionsystem	50
<a href="#">util::IPlatform</a>	52
<a href="#">LevelUpInstance</a>	54
<a href="#">LevelUpSystem</a>	55
<a href="#">util::LinuxPlatform</a>	58
<a href="#">Map</a>	59
<a href="#">Monster</a>	
<a href="#">Monster</a> class, which all our other monsters inherit	63
<a href="#">MonsterSpawner</a>	67

<a href="#">Collision::OrientedBoundingBox</a>	69
<a href="#">Player</a>	
<a href="#">Player</a> class, our controlled character	70
<a href="#">Potion</a>	75
<a href="#">PowerUp</a>	77
<a href="#">Projectile</a>	78
<a href="#">RandomMonster</a>	
<a href="#">RandomMonster</a> is a monster that moves randomly around and shoots projectiles towards the player	85
<a href="#">RedPotion</a>	
A red potion. Takes a position as parameter and the colour and healing effect is set automatically	87
<a href="#">RoomInstance</a>	
Class representing a room of a dungeon, usually includes a monsterspawner	88
<a href="#">RoomTile</a>	97
<a href="#">ScreenText</a>	99
<a href="#">SearchingMonster</a>	
<a href="#">SearchingMonster</a> is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player	100
<a href="#">SlowMonster</a>	
<a href="#">SlowMonster</a> is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposely inaccurate	102
<a href="#">SnipingMonster</a>	
<a href="#">SnipingMonster</a> never moves but shoots projectile towards the player with a really big range	104
<a href="#">SoundEffect</a>	106
<a href="#">StartingRoom</a>	107
<a href="#">SwordWeapon</a>	108
<a href="#">TreasureRoom</a>	
Unused, was going to represent a room that includes a treasure and no monsters	109
<a href="#">VioletPotion</a>	
A violet potion. Takes a position as parameter and the colour and healing effect is set automatically	111
<a href="#">WallPatrolMonster</a>	
<a href="#">WallPatrolMonster</a> walks along the walls of the room patrolling. Then when the player is in its range it shoots projectiles towards it	112
<a href="#">WallTile</a>	114
<a href="#">Weapon</a>	114
<a href="#">util::WindowsPlatform</a>	119
<a href="#">YellowPotion</a>	
A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically	121

## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">entity.cpp</a>	156
src/ <a href="#">entity.hpp</a>	156
src/ <a href="#">game.cpp</a>	157
src/ <a href="#">game.hpp</a>	158
src/ <a href="#">gamebar.cpp</a>	159
src/ <a href="#">gamebar.hpp</a>	160
src/ <a href="#">PCH.hpp</a>	163
src/Actors/ <a href="#">character.cpp</a>	123
src/Actors/ <a href="#">character.hpp</a>	124
src/Actors/ <a href="#">player.cpp</a>	137
src/Actors/ <a href="#">player.hpp</a>	137
src/Actors/Monsters/ <a href="#">BossMonster.cpp</a>	126
src/Actors/Monsters/ <a href="#">BossMonster.hpp</a>	126
src/Actors/Monsters/ <a href="#">monster.cpp</a>	127
src/Actors/Monsters/ <a href="#">monster.hpp</a>	127
src/Actors/Monsters/ <a href="#">RandomMonster.cpp</a>	131
src/Actors/Monsters/ <a href="#">RandomMonster.hpp</a>	131
src/Actors/Monsters/ <a href="#">SearchingMonster.cpp</a>	132
src/Actors/Monsters/ <a href="#">SearchingMonster.hpp</a>	132
src/Actors/Monsters/ <a href="#">SlowMonster.cpp</a>	133
src/Actors/Monsters/ <a href="#">SlowMonster.hpp</a>	133
src/Actors/Monsters/ <a href="#">SnipingMonster.cpp</a>	134
src/Actors/Monsters/ <a href="#">SnipingMonster.hpp</a>	135
src/Actors/Monsters/ <a href="#">WallPatrolMonster.cpp</a>	136
src/Actors/Monsters/ <a href="#">WallPatrolMonster.hpp</a>	136
src/Actors/Monsters/MonsterSpawner/ <a href="#">BossSpawner.cpp</a>	128
src/Actors/Monsters/MonsterSpawner/ <a href="#">BossSpawner.hpp</a>	128
src/Actors/Monsters/MonsterSpawner/ <a href="#">MonsterSpawner.cpp</a>	129
src/Actors/Monsters/MonsterSpawner/ <a href="#">MonsterSpawner.hpp</a>	129
src/Animation/ <a href="#">animation.cpp</a>	138
src/Animation/ <a href="#">animation.hpp</a>	138
src/Animation/ <a href="#">Animationhandler.cpp</a>	139
src/Animation/ <a href="#">Animationhandler.hpp</a>	139
src/Combat/ <a href="#">CircularProjectile.hpp</a>	141
src/Combat/ <a href="#">projectile.cpp</a>	144

src/Combat/Projectile.hpp	144
src/Combat/Health/HealthPotions.hpp	141
src/Combat/Health/Potion.cpp	142
src/Combat/Health/Potion.hpp	143
src/Combat/PowerUps/PowerUp.hpp	143
src/Combat/Weapons/BowWeapon.cpp	146
src/Combat/Weapons/BowWeapon.hpp	146
src/Combat/Weapons/SwordWeapon.cpp	147
src/Combat/Weapons/SwordWeapon.hpp	147
src/Combat/Weapons/Weapon.cpp	148
src/Combat/Weapons/Weapon.hpp	148
src/Dungeon/map.cpp	149
src/Dungeon/map.hpp	149
src/Dungeon/roomInstance.cpp	151
src/Dungeon/roomInstance.hpp	151
src/Dungeon/specialrooms/BossRoom.cpp	153
src/Dungeon/specialrooms/BossRoom.hpp	153
src/Dungeon/specialrooms/startingRoom.cpp	154
src/Dungeon/specialrooms/startingRoom.hpp	154
src/Dungeon/specialrooms/TreasureRoom.cpp	154
src/Dungeon/specialrooms/TreasureRoom.hpp	155
src/Dungeon/Tiles/roomTile.cpp	155
src/Dungeon/Tiles/roomTile.hpp	155
src/Interfaces/CollisionSystem.hpp	161
src/Interfaces/ICollidable.hpp	162
src/Platform/IPlatform.hpp	164
src/Platform/Platform.hpp	165
src/Platform/Unix/LinuxPlatform.cpp	165
src/Platform/Unix/LinuxPlatform.hpp	165
src/Platform/Win32/Resource.h	166
src/Platform/Win32/WindowsPlatform.cpp	167
src/Platform/Win32/WindowsPlatform.hpp	167
src/Utility/Collision.cpp	168
src/Utility/Collision.hpp	168
src/Utility/FileSystem.hpp	169
src/Utility/LevelUpInstance.cpp	170
src/Utility/LevelUpInstance.hpp	170
src/Utility/LevelUpSystem.cpp	171
src/Utility/LevelUpSystem.hpp	171
src/Utility/Main.cpp	172
src/Utility/RandomHelper.cpp	172
src/Utility/RandomHelper.hpp	173
src/Utility/ScreenText.cpp	173
src/Utility/ScreenText.hpp	174
src/Utility/SpriteHelper.cpp	175
src/Utility/SpriteHelper.hpp	175
src/Utility/Types.hpp	176
src/Utility/Sounds/SoundEffects.cpp	174
src/Utility/Sounds/SoundEffects.hpp	174



## Chapter 6

# Namespace Documentation

### 6.1 Collision Namespace Reference

#### Classes

- class [BitmaskManager](#)
- class [OrientedBoundingBox](#)

#### Functions

- bool [PixelPerfectTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2, sf::Uint8 AlphaLimit)
- bool [CreateTextureAndBitmask](#) (sf::Texture &LoadInto, const std::string &Filename)
- sf::Vector2f [GetSpriteCenter](#) (const sf::Sprite &Object)
- sf::Vector2f [GetSpriteSize](#) (const sf::Sprite &Object)
- bool [CircleTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2)
- bool [BoundingBoxTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2)

#### Variables

- [BitmaskManager](#) Bitmasks

#### 6.1.1 Function Documentation

##### 6.1.1.1 BoundingBoxTest()

```
bool Collision::BoundingBoxTest (
    const sf::Sprite & Object1,
    const sf::Sprite & Object2 )
```

Test for bounding box collision using the Separating Axis Theorem Supports scaling and rotation

#### 6.1.1.2 CircleTest()

```
bool Collision::CircleTest (
    const sf::Sprite & Object1,
    const sf::Sprite & Object2 )
```

Test for collision using circle collision detection Radius is averaged from the dimensions of the sprite so roughly circular objects will be much more accurate

#### 6.1.1.3 CreateTextureAndBitmask()

```
bool Collision::CreateTextureAndBitmask (
    sf::Texture & LoadInto,
    const std::string & Filename )
```

Replaces Texture::loadFromFile Load an imagefile into the given texture and create a bitmask for it This is much faster than creating the bitmask for a texture on the first run of "PixelPerfectTest"

The function returns false if the file could not be opened for some reason

#### 6.1.1.4 GetSpriteCenter()

```
sf::Vector2f Collision::GetSpriteCenter (
    const sf::Sprite & Object )
```

#### 6.1.1.5 GetSpriteSize()

```
sf::Vector2f Collision::GetSpriteSize (
    const sf::Sprite & Object )
```

#### 6.1.1.6 PixelPerfectTest()

```
bool Collision::PixelPerfectTest (
    const sf::Sprite & Object1,
    const sf::Sprite & Object2,
    sf::Uint8 AlphaLimit = 0 )
```

Test for a collision between two sprites by comparing the alpha values of overlapping pixels Supports scaling and rotation AlphaLimit: The threshold at which a pixel becomes "solid". If AlphaLimit is 127, a pixel with alpha value 128 will cause a collision and a pixel with alpha value 126 will not.

This functions creates bitmasks of the textures of the two sprites by downloading the textures from the graphics card to memory -> SLOW! You can avoid this by using the "CreateTextureAndBitmask" function

## 6.1.2 Variable Documentation

### 6.1.2.1 Bitmasks

`BitmaskManager Collision::Bitmasks`

## 6.2 direction Namespace Reference

### Functions

- `Direction GetOppositeDir (Direction direction)`

### 6.2.1 Function Documentation

#### 6.2.1.1 GetOppositeDir()

```
Direction direction::GetOppositeDir (  
    Direction direction )
```

## 6.3 randomhelper Namespace Reference

### Functions

- float `RandomFloatBetween` (float min, float max)
- int `RandomIntBetween` (int min, int max)

### 6.3.1 Function Documentation

#### 6.3.1.1 RandomFloatBetween()

```
float randomhelper::RandomFloatBetween (  
    float min,  
    float max )
```

### 6.3.1.2 RandomIntBetween()

```
int randomhelper::RandomIntBetween (
    int min,
    int max )
```

## 6.4 spritehelper Namespace Reference

### Functions

- void [CreateSpriteFrom](#) (const std::string &spriteFile, sf::Vector2f dimensions, sf::Sprite &sprite, sf::Texture &texture)
- void [SetScale](#) (sf::Vector2f wantedDimension, sf::Sprite &sprite)
- void [RotateSprite](#) (sf::Vector2f directionOfRotation, sf::Sprite &sprite)
- void [SetOriginBottomCenter](#) (sf::Sprite &sprite)

### 6.4.1 Function Documentation

#### 6.4.1.1 CreateSpriteFrom()

```
void spritehelper::CreateSpriteFrom (
    const std::string & spriteFile,
    sf::Vector2f dimensions,
    sf::Sprite & sprite,
    sf::Texture & texture )
```

#### 6.4.1.2 RotateSprite()

```
void spritehelper::RotateSprite (
    sf::Vector2f directionOfRotation,
    sf::Sprite & sprite )
```

#### 6.4.1.3 SetOriginBottomCenter()

```
void spritehelper::SetOriginBottomCenter (
    sf::Sprite & sprite )
```

#### 6.4.1.4 SetScale()

```
void spritehelper::SetScale (
    sf::Vector2f wantedDimension,
    sf::Sprite & sprite )
```

## 6.5 util Namespace Reference

### Classes

- struct [IPlatform](#)
- struct [LinuxPlatform](#)
- struct [WindowsPlatform](#)



## Chapter 7

# Class Documentation

### 7.1 Animation Class Reference

[Animation](#) build the characters animation frames from spritesheet and displays them to achive an animated sprite.

```
#include <animation.hpp>
```

#### Public Member Functions

- [Animation](#) ()=default
- [Animation](#) (int x, int y, int width, int height, int spacing, const std::string &textureName)  
*Construct a new [Animation](#) object.*
- [~Animation](#) ()
- void [AnimationToSprite](#) (sf::Sprite &sprite) const  
*sets the right texture to sprite*
- void [Update](#) (float dt)  
*updates the sprite based on time passed*

#### 7.1.1 Detailed Description

[Animation](#) build the characters animation frames from spritesheet and displays them to achive an animated sprite.

#### 7.1.2 Constructor & Destructor Documentation

##### 7.1.2.1 Animation() [1/2]

```
Animation::Animation ( ) [default]
```

### 7.1.2.2 Animation() [2/2]

```
Animation::Animation (
    int x,
    int y,
    int width,
    int height,
    int spacing,
    const std::string & textureName )
```

Construct a new [Animation](#) object.

#### Parameters

<i>x</i>	x-coordinate of rectangle defining the sprite in spritesheet
<i>y</i>	y-coordinate of rectangle defining the sprite in spritesheet
<i>width</i>	width of the character in animation
<i>height</i>	height of the character in animation
<i>textureName</i>	the spritesheet animation pulls from to create the animation

### 7.1.2.3 ~Animation()

```
Animation::~~Animation ( ) [inline]
```

## 7.1.3 Member Function Documentation

### 7.1.3.1 AnimationToSprite()

```
void Animation::AnimationToSprite (
    sf::Sprite & sprite ) const
```

sets the right texture to sprite

#### Parameters

<i>sprite</i>	sprite which the texture is added to
---------------	--------------------------------------

### 7.1.3.2 Update()

```
void Animation::Update (
    float dt )
```



updates the sprite based on time passed

## Parameters

<i>dt</i>	time parameter
-----------	----------------

The documentation for this class was generated from the following files:

- src/Animation/[animation.hpp](#)
- src/Animation/[animation.cpp](#)

## 7.2 AnimationHandler Class Reference

```
#include <Animationhandler.hpp>
```

### Public Member Functions

- [AnimationHandler](#) ()
- [~AnimationHandler](#) ()
- [AnimationHandler](#) (uint xOffset, uint yOffset, uint width, uint height, uint xSpacing, const std::string &textureLocation, const std::string &deathTexture)
- void [setAnimation](#) ([AnimationIndex](#) index)
- [Animation](#) \* [getAnimation](#) () const

### 7.2.1 Constructor & Destructor Documentation

#### 7.2.1.1 AnimationHandler() [1/2]

```
AnimationHandler::AnimationHandler ( ) [inline]
```

#### 7.2.1.2 ~AnimationHandler()

```
AnimationHandler::~~AnimationHandler ( )
```

#### 7.2.1.3 AnimationHandler() [2/2]

```
AnimationHandler::AnimationHandler (
    uint xOffset,
    uint yOffset,
    uint width,
    uint height,
    uint xSpacing,
    const std::string & textureLocation,
    const std::string & deathTexture )
```

## 7.2.2 Member Function Documentation

### 7.2.2.1 getAnimation()

```
Animation * AnimationHandler::getAnimation ( ) const
```

### 7.2.2.2 setAnimation()

```
void AnimationHandler::setAnimation (
    AnimationIndex index )
```

The documentation for this class was generated from the following files:

- src/Animation/[Animationhandler.hpp](#)
- src/Animation/[Animationhandler.cpp](#)

## 7.3 Collision::BitmaskManager Class Reference

### Public Member Functions

- [~BitmaskManager](#) ( )
- sf::Uint8 [GetPixel](#) (const sf::Uint8 \*mask, const sf::Texture \*tex, unsigned int x, unsigned int y)
- sf::Uint8 \* [GetMask](#) (const sf::Texture \*tex)
- sf::Uint8 \* [CreateMask](#) (const sf::Texture \*tex, const sf::Image &img)

### 7.3.1 Constructor & Destructor Documentation

#### 7.3.1.1 ~BitmaskManager()

```
Collision::BitmaskManager::~~BitmaskManager ( ) [inline]
```

### 7.3.2 Member Function Documentation

### 7.3.2.1 CreateMask()

```
sf::Uint8 * Collision::BitmaskManager::CreateMask (
    const sf::Texture * tex,
    const sf::Image & img ) [inline]
```

### 7.3.2.2 GetMask()

```
sf::Uint8 * Collision::BitmaskManager::GetMask (
    const sf::Texture * tex ) [inline]
```

### 7.3.2.3 GetPixel()

```
sf::Uint8 Collision::BitmaskManager::GetPixel (
    const sf::Uint8 * mask,
    const sf::Texture * tex,
    unsigned int x,
    unsigned int y ) [inline]
```

The documentation for this class was generated from the following file:

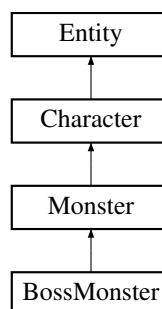
- src/Utility/[Collision.cpp](#)

## 7.4 BossMonster Class Reference

This is the boss monster. It is the most powerful monster and once the player kills it the game is won.

```
#include <BossMonster.hpp>
```

Inheritance diagram for BossMonster:



## Public Member Functions

- [BossMonster](#) ([PlayerPS](#) player, float xPos, float yPos)
- [BossMonster](#) ([PlayerPS](#) player, [sf::Vector2f](#) pos)
- [~BossMonster](#) ()
- virtual [std::list< ProjectileUP >](#) [Attack](#) ()
- virtual bool [Move](#) (float dt)
- void [initVariables](#) ()

## Additional Inherited Members

### 7.4.1 Detailed Description

This is the boss monster. It is the most powerful monster and once the player kills it the game is won.

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 [BossMonster\(\)](#) [1/2]

```
BossMonster::BossMonster (  
    PlayerPS player,  
    float xPos,  
    float yPos )
```

#### 7.4.2.2 [BossMonster\(\)](#) [2/2]

```
BossMonster::BossMonster (  
    PlayerPS player,  
    sf::Vector2f pos )
```

#### 7.4.2.3 [~BossMonster\(\)](#)

```
BossMonster::~~BossMonster ( )
```

### 7.4.3 Member Function Documentation

#### 7.4.3.1 Attack()

```
std::list< ProjectileUP > BossMonster::Attack ( ) [virtual]
```

Implements [Monster](#).

#### 7.4.3.2 initVariables()

```
void BossMonster::initVariables ( )
```

#### 7.4.3.3 Move()

```
bool BossMonster::Move (
    float dt ) [virtual]
```

Implements [Monster](#).

The documentation for this class was generated from the following files:

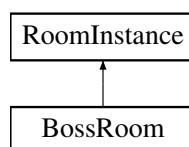
- [src/Actors/Monsters/BossMonster.hpp](#)
- [src/Actors/Monsters/BossMonster.cpp](#)

## 7.5 BossRoom Class Reference

The roominstance where the boss spawns and which has to be cleared to beat the game.

```
#include <BossRoom.hpp>
```

Inheritance diagram for BossRoom:



### Public Member Functions

- [BossRoom](#) (sf::Vector2u window\_size, sf::Vector2i coords)
- [BossRoom](#) ()
- [~BossRoom](#) ()
- virtual void [Enter](#) ([PlayerPS](#) player, [Direction](#) direction)
- virtual void [setTiles](#) (sf::Vector2u window\_size)

## Additional Inherited Members

### 7.5.1 Detailed Description

The roominstance where the boss spawns and which has to be cleared to beat the game.

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 BossRoom() [1/2]

```
BossRoom::BossRoom (
    sf::Vector2u window_size,
    sf::Vector2i coords )
```

See also

[RoomInstance::RoomInstance\(sf::Vector2u window\\_size, sf::Vector2i coords\)](#)

#### 7.5.2.2 BossRoom() [2/2]

```
BossRoom::BossRoom ( ) [inline]
```

#### 7.5.2.3 ~BossRoom()

```
BossRoom::~BossRoom ( ) [inline]
```

### 7.5.3 Member Function Documentation

#### 7.5.3.1 Enter()

```
void BossRoom::Enter (
    PlayerPS player,
    Direction direction ) [virtual]
```

## Parameters

<i>player</i>	Description
<i>direction</i>	Description

Reimplemented from [RoomInstance](#).

### 7.5.3.2 setTiles()

```
void BossRoom::setTiles (
    sf::Vector2u window_size ) [virtual]
```

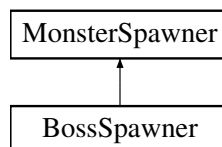
The documentation for this class was generated from the following files:

- [src/Dungeon/specialrooms/BossRoom.hpp](#)
- [src/Dungeon/specialrooms/BossRoom.cpp](#)

## 7.6 BossSpawner Class Reference

```
#include <BossSpawner.hpp>
```

Inheritance diagram for BossSpawner:



### Public Member Functions

- [BossSpawner](#) ()
- [~BossSpawner](#) ()
- virtual [MonsterSP SpawnMonster](#) (sf::Vector2u roomSize, [PlayerPS](#) target)

### Additional Inherited Members

#### 7.6.1 Constructor & Destructor Documentation



### 7.6.1.1 BossSpawner()

```
BossSpawner::BossSpawner ( ) [inline]
```

### 7.6.1.2 ~BossSpawner()

```
BossSpawner::~~BossSpawner ( ) [inline]
```

## 7.6.2 Member Function Documentation

### 7.6.2.1 SpawnMonster()

```
MonsterSP BossSpawner::SpawnMonster (
    sf::Vector2u roomSize,
    PlayerPS target ) [virtual]
```

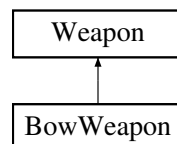
The documentation for this class was generated from the following files:

- src/Actors/Monsters/MonsterSpawner/[BossSpawner.hpp](#)
- src/Actors/Monsters/MonsterSpawner/[BossSpawner.cpp](#)

## 7.7 BowWeapon Class Reference

```
#include <BowWeapon.hpp>
```

Inheritance diagram for BowWeapon:



### Public Member Functions

- [BowWeapon](#) (int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const std::string &spriteLocation)
- virtual [~BowWeapon](#) ()
- virtual [ProjectileUP Use](#) (Vector2f dir, Vector2f origin)

## Additional Inherited Members

### 7.7.1 Constructor & Destructor Documentation

#### 7.7.1.1 BowWeapon()

```
BowWeapon::BowWeapon (
    int damage,
    int range,
    int rateOfFire,
    float projectileSpeed,
    Vector2f projectileSize,
    const std::string & spriteLocation )
```

#### 7.7.1.2 ~BowWeapon()

```
virtual BowWeapon::~BowWeapon ( ) [inline], [virtual]
```

### 7.7.2 Member Function Documentation

#### 7.7.2.1 Use()

```
ProjectileUP BowWeapon::Use (
    Vector2f dir,
    Vector2f origin ) [virtual]
```

Implements [Weapon](#).

The documentation for this class was generated from the following files:

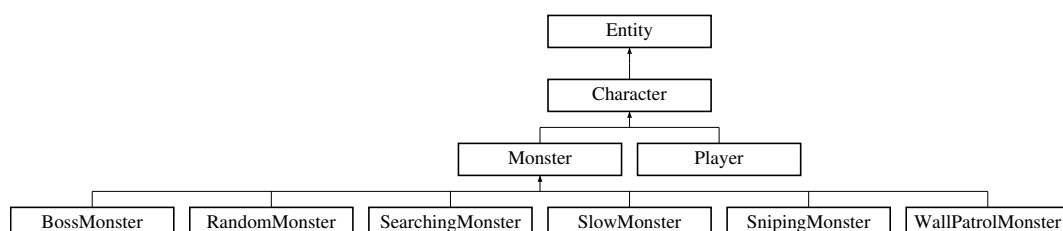
- [src/Combat/Weapons/BowWeapon.hpp](#)
- [src/Combat/Weapons/BowWeapon.cpp](#)

## 7.8 Character Class Reference

[Character](#) class that our player and monster inherits.

```
#include <character.hpp>
```

Inheritance diagram for Character:



## Public Member Functions

- [Character](#) (const std::string &filename, sf::Vector2f pos, bool animated=false)
- virtual [~Character](#) ()
- virtual void [Update](#) (float dt)=0
- void [Equip](#) ([Weapon](#) \*weapon)  
*Equips weapon to character.*
- void [initVariables](#) ()
- void [TakeDamage](#) (int value)  
*removes hitpoints corresponding to the damage taken value*
- void [Heal](#) (int value)  
*Heal character.*
- int [GetHitPoints](#) () const
- bool [IsAlive](#) ()  
*Check if player is alive.*
- bool [HasWeapon](#) ()  
*Check if player has weapon equipped.*
- bool [Idle](#) ()  
*Set animation to IDLE.*
- bool [Dead](#) ()  
*Set animation to DEAD.*
- bool [MoveLeft](#) (float dt)  
*Sets the animation to right direction (LEFT) and moves the character.*
- bool [MoveRight](#) (float dt)  
*Sets the animation to right direction (RIGHT) and moves the character.*
- bool [MoveDown](#) (float dt)  
*Sets the animation to right direction (DOWN) and moves the character.*
- bool [MoveUp](#) (float dt)  
*Sets the animation to right direction (UP) and moves the character.*
- virtual bool [Move](#) (float)
- void [RevertMove](#) ()  
*revert movement to last position, used when hit wall*
- void [ResetAttackCooldown](#) ()
- float [GetAttackCooldownLeft](#) () const
- float [GetAttackCooldownLength](#) () const
- int [GetMaxHP](#) ()
- void [SetNormalSpeed](#) (float value)
- void [ResetCharacterToBeAlive](#) ()

## Public Attributes

- bool [CanAttack](#)

## Protected Member Functions

- void [generalUpdate](#) (float dt)
- void [updateAttackCooldown](#) (float dt)
- std::list< [ProjectileUP](#) > [emptyList](#) ()
- std::list< [ProjectileUP](#) > [shotProjectileList](#) (sf::Vector2f aimPos)

## Protected Attributes

- [Weapon](#) \* [weapon\\_](#)
- [Projectile::Type](#) [characterProjectileType\\_](#)
- [sf::Vector2f](#) [startPos](#)
- [int](#) [hitpoints\\_](#)
- [int](#) [currentMaxHitpoints\\_](#)
- [int](#) [defaultMaxHitpoints\\_](#)
- [bool](#) [hasAnimation\\_](#)
- [AnimationHandler](#) [animationHandler\\_](#)
- [float](#) [currentSpeed\\_](#)
- [float](#) [defaultSpeed\\_](#)
- [bool](#) [left\\_or\\_right\\_](#) = true
- [bool](#) [invincibility\\_frame\\_](#) = false
- [float](#) [attackCooldownLength\\_](#)
- [float](#) [attackCooldownLeft](#)

### 7.8.1 Detailed Description

[Character](#) class that our player and monster inherits.

### 7.8.2 Constructor & Destructor Documentation

#### 7.8.2.1 Character()

```
Character::Character (
    const std::string & filename,
    sf::Vector2f pos,
    bool animated = false )
```

#### 7.8.2.2 ~Character()

```
Character::~~Character ( ) [virtual]
```

### 7.8.3 Member Function Documentation

### 7.8.3.1 Dead()

```
bool Character::Dead ( )
```

Set animation to DEAD.

#### Returns

true when set

### 7.8.3.2 emptyList()

```
std::list< ProjectileUP > Character::emptyList ( ) [protected]
```

### 7.8.3.3 Equip()

```
void Character::Equip (
    Weapon * weapon )
```

Equips weapon to character.

#### Parameters

<i>weapon</i>	weapon to be equipped
---------------	-----------------------

### 7.8.3.4 generalUpdate()

```
void Character::generalUpdate (
    float dt ) [protected]
```

### 7.8.3.5 GetAttackCooldownLeft()

```
float Character::GetAttackCooldownLeft ( ) const [inline]
```

### 7.8.3.6 GetAttackCooldownLength()

```
float Character::GetAttackCooldownLength ( ) const [inline]
```

#### 7.8.3.7 GetHitPoints()

```
int Character::GetHitPoints ( ) const
```

#### 7.8.3.8 GetMaxHP()

```
int Character::GetMaxHP ( )
```

#### 7.8.3.9 HasWeapon()

```
bool Character::HasWeapon ( )
```

Check if player has weapon equipped.

##### Returns

true if has  
false if not

#### 7.8.3.10 Heal()

```
void Character::Heal (
    int value )
```

Heal character.

##### Parameters

<i>value</i>	value healed
--------------	--------------

#### 7.8.3.11 Idle()

```
bool Character::Idle ( )
```

Set animation to IDLE.

##### Returns

true when set

### 7.8.3.12 initVariables()

```
void Character::initVariables ( )
```

### 7.8.3.13 IsAlive()

```
bool Character::IsAlive ( )
```

Check if player is alive.

#### Returns

true if alive  
false if not

### 7.8.3.14 Move()

```
virtual bool Character::Move (
    float ) [inline], [virtual]
```

Reimplemented in [BossMonster](#), [RandomMonster](#), [SearchingMonster](#), [SlowMonster](#), [SnipingMonster](#), [WallPatrolMonster](#), and [Monster](#).

### 7.8.3.15 MoveDown()

```
bool Character::MoveDown (
    float dt )
```

Sets the animation to right direction (DOWN) and moves the character.

#### Parameters

<i>dt</i>	deltatime
-----------	-----------

#### Returns

true when set

### 7.8.3.16 MoveLeft()

```
bool Character::MoveLeft (
    float dt )
```

Sets the animation to right direction (LEFT) and moves the character.

#### Parameters

<i>dt</i>	deltatime
-----------	-----------

#### Returns

true when set

```
else if (!left_or_right_) { sprite_.setScale(-sprite_.getScale().x, sprite_.getScale().y); sprite_.setPosition(pos_.x +
sprite_.getLocalBounds().width, pos_.y); left_or_right_ = true; }
```

### 7.8.3.17 MoveRight()

```
bool Character::MoveRight (
    float dt )
```

Sets the animation to right direction (RIGHT) and moves the character.

#### Parameters

<i>dt</i>	deltatime
-----------	-----------

#### Returns

true when set

```
else if (left_or_right_) { sprite_.setScale(sprite_.getScale().x, sprite_.getScale().y); sprite_.setPosition(pos_.x + 64,
pos_.y); left_or_right_ = false; }
```

### 7.8.3.18 MoveUp()

```
bool Character::MoveUp (
    float dt )
```

Sets the animation to right direction (UP) and moves the character.

#### Parameters

<i>dt</i>	deltatime
-----------	-----------



**Returns**

true when set

**7.8.3.19 ResetAttackCooldown()**

```
void Character::ResetAttackCooldown ( )
```

**7.8.3.20 ResetCharacterToBeAlive()**

```
void Character::ResetCharacterToBeAlive ( )
```

**7.8.3.21 RevertMove()**

```
void Character::RevertMove ( )
```

revert movement to last position, used when hit wall

**7.8.3.22 SetNormalSpeed()**

```
void Character::SetNormalSpeed (
    float value )
```

**7.8.3.23 shotProjectileList()**

```
std::list< ProjectileUP > Character::shotProjectileList (
    sf::Vector2f aimPos ) [protected]
```

**7.8.3.24 TakeDamage()**

```
void Character::TakeDamage (
    int value )
```

removes hitpoints corresponding to the damage taken value

**Parameters**

<i>value</i>	damage taken
--------------	--------------

**7.8.3.25 Update()**

```
virtual void Character::Update (  
    float dt ) [pure virtual]
```

Implemented in [Player](#), and [Monster](#).

**7.8.3.26 updateAttackCooldown()**

```
void Character::updateAttackCooldown (  
    float dt ) [protected]
```

**7.8.4 Member Data Documentation****7.8.4.1 animationHandler\_**

[AnimationHandler](#) Character::animationHandler\_ [protected]

**7.8.4.2 attackCooldownLeft**

float Character::attackCooldownLeft [protected]

**7.8.4.3 attackCooldownLength\_**

float Character::attackCooldownLength\_ [protected]

#### 7.8.4.4 CanAttack

```
bool Character::CanAttack
```

#### 7.8.4.5 characterProjectileType\_

```
Projectile::Type Character::characterProjectileType_ [protected]
```

#### 7.8.4.6 currentMaxHitpoints\_

```
int Character::currentMaxHitpoints_ [protected]
```

#### 7.8.4.7 currentSpeed\_

```
float Character::currentSpeed_ [protected]
```

#### 7.8.4.8 defaultMaxHitpoints\_

```
int Character::defaultMaxHitpoints_ [protected]
```

#### 7.8.4.9 defaultSpeed\_

```
float Character::defaultSpeed_ [protected]
```

#### 7.8.4.10 hasAnimation\_

```
bool Character::hasAnimation_ [protected]
```

#### 7.8.4.11 hitpoints\_

```
int Character::hitpoints_ [protected]
```

#### 7.8.4.12 invincibility\_frame\_

```
bool Character::invincibility_frame_ = false [protected]
```

#### 7.8.4.13 left\_or\_right\_

```
bool Character::left_or_right_ = true [protected]
```

#### 7.8.4.14 startPos

```
sf::Vector2f Character::startPos [protected]
```

#### 7.8.4.15 weapon\_

```
Weapon* Character::weapon_ [protected]
```

The documentation for this class was generated from the following files:

- src/Actors/[character.hpp](#)
- src/Actors/[character.cpp](#)

## 7.9 CollisionSystem Class Reference

Unused, was not intergrated or completed.

```
#include <CollisionSystem.hpp>
```

### Public Member Functions

- [CollisionSystem](#) ()
- void [AddToCollisionList](#) ([ICollidable](#) \*obj)
- void [RemoveFromCollisionList](#) ([ICollidable](#) \*obj)
- void [ProcessCollisionList](#) ()

#### 7.9.1 Detailed Description

Unused, was not intergrated or completed.

## 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 CollisionSystem()

```
CollisionSystem::CollisionSystem ( ) [inline]
```

## 7.9.3 Member Function Documentation

### 7.9.3.1 AddToCollisionList()

```
void CollisionSystem::AddToCollisionList (
    ICollidable * obj )
```

### 7.9.3.2 ProcessCollisionList()

```
void CollisionSystem::ProcessCollisionList ( )
```

### 7.9.3.3 RemoveFromCollisionList()

```
void CollisionSystem::RemoveFromCollisionList (
    ICollidable * obj )
```

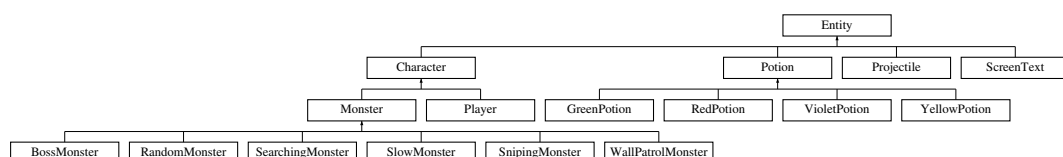
The documentation for this class was generated from the following file:

- [src/Interfaces/CollisionSystem.hpp](#)

## 7.10 Entity Class Reference

```
#include <entity.hpp>
```

Inheritance diagram for Entity:



## Public Member Functions

- [Entity](#) (const std::string &spriteLocation, float xPos, float yPos, sf::Vector2f spriteDims)  
*Construct a new game [Entity](#).*
- [Entity](#) (const std::string &spirteLocation, sf::Vector2f pos, sf::Vector2f spriteDims)
- [Entity](#) (sf::Sprite &sprite, float xPos, float yPos)
- [Entity](#) (sf::Sprite &sprite, sf::Vector2f pos)
- virtual [~Entity](#) ()
- const sf::Sprite & [GetSprite](#) () const  
*Get the entity's sprite.*
- const sf::Vector2f & [GetPos](#) () const  
*Get the entity position as a [Vector2f](#).*
- const sf::Vector2i [GetPosI](#) ()  
*Get the entity postions.*
- sf::Vector2f [GetSpritePosition](#) () const  
*Get the position of the entitys sprite.*
- sf::Vector2f [GetSpriteCenter](#) () const  
*Get the position of the entitys sprite center.*
- const sf::Vector2f & [GetOldPosition](#) () const  
*Get the position of the Entitys position during the last game tick.*
- sf::FloatRect [GetSpriteBounds](#) () const
- sf::FloatRect [GetBaseBoxAt](#) (sf::Vector2f pos) const
- void [SetPos](#) (sf::Vector2f pos)  
*Function to set the position and sprite position of the entity.*
- void [SetPosAndOldPos](#) (sf::Vector2f pos)  
*Function to set both the oldPos\_, pos\_ and sprite position of the entity.*
- virtual void [Render](#) (sf::RenderTarget \*target)

## Protected Member Functions

- void [initSprite](#) (const std::string &spriteLocation, sf::Vector2f spriteDims)

## Protected Attributes

- sf::Vector2f [pos\\_](#)
- sf::Vector2f [oldPos\\_](#)
- sf::Sprite [sprite\\_](#)
- sf::Texture [texture\\_](#)

## 7.10.1 Constructor & Destructor Documentation

### 7.10.1.1 Entity() [1/4]

```
Entity::Entity (
    const std::string & spriteLocation,
    float xPos,
    float yPos,
    sf::Vector2f spriteDims )
```

Construct a new game [Entity](#).

## Parameters

<i>spriteLocation</i>	path to the sprite
<i>xPos</i>	x-choord for left side
<i>yPos</i>	y-choord for top
<i>spriteDims</i>	!scaling of sprite!

**7.10.1.2 Entity()** [2/4]

```
Entity::Entity (
    const std::string & spirteLocation,
    sf::Vector2f pos,
    sf::Vector2f spriteDims )
```

**7.10.1.3 Entity()** [3/4]

```
Entity::Entity (
    sf::Sprite & sprite,
    float xPos,
    float yPos )
```

**7.10.1.4 Entity()** [4/4]

```
Entity::Entity (
    sf::Sprite & sprite,
    sf::Vector2f pos )
```

**7.10.1.5 ~Entity()**

```
virtual Entity::~Entity ( ) [inline], [virtual]
```

**7.10.2 Member Function Documentation**

#### 7.10.2.1 GetBaseBoxAt()

```
sf::FloatRect Entity::GetBaseBoxAt (
    sf::Vector2f pos ) const
```

#### 7.10.2.2 GetOldPosition()

```
const sf::Vector2f & Entity::GetOldPosition ( ) const
```

Get the position of the Entity's position during the last game tick.

##### Returns

const sf::Vector2f&

#### 7.10.2.3 GetPos()

```
const sf::Vector2f & Entity::GetPos ( ) const [inline]
```

Get the entity position as a Vector2f.

##### Returns

const sf::Vector2f&

#### 7.10.2.4 GetPosI()

```
const sf::Vector2i Entity::GetPosI ( ) [inline]
```

Get the entity positions.

##### Returns

const sf::Vector2i



### 7.10.2.5 GetSprite()

```
const sf::Sprite & Entity::GetSprite ( ) const [inline]
```

Get the entity's sprite.

#### Returns

const sf::Sprite&

### 7.10.2.6 GetSpriteBounds()

```
sf::FloatRect Entity::GetSpriteBounds ( ) const
```

### 7.10.2.7 GetSpriteCenter()

```
sf::Vector2f Entity::GetSpriteCenter ( ) const
```

Get the position of the entity's sprite center.

#### Returns

sf::Vector2f

### 7.10.2.8 GetSpritePosition()

```
sf::Vector2f Entity::GetSpritePosition ( ) const [inline]
```

Get the position of the entity's sprite.

#### Returns

sf::Vector2f

### 7.10.2.9 initSprite()

```
void Entity::initSprite (
    const std::string & spriteLocation,
    sf::Vector2f spriteDims ) [protected]
```

#### 7.10.2.10 Render()

```
void Entity::Render (
    sf::RenderTarget * target ) [virtual]
```

Reimplemented in [Monster](#).

#### 7.10.2.11 SetPos()

```
void Entity::SetPos (
    sf::Vector2f pos )
```

Function to set the position and sprite position of the entity.

##### Parameters

<i>pos</i>	the new position
------------	------------------

#### 7.10.2.12 SetPosAndOldPos()

```
void Entity::SetPosAndOldPos (
    sf::Vector2f pos )
```

Function to set both the oldPos\_, pos\_ and sprite position of the entity.

##### Parameters

<i>pos</i>	the new position
------------	------------------

### 7.10.3 Member Data Documentation

#### 7.10.3.1 oldPos\_

```
sf::Vector2f Entity::oldPos_ [protected]
```

#### 7.10.3.2 pos\_

```
sf::Vector2f Entity::pos_ [protected]
```

### 7.10.3.3 sprite\_

```
sf::Sprite Entity::sprite_ [protected]
```

### 7.10.3.4 texture\_

```
sf::Texture Entity::texture_ [protected]
```

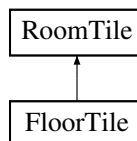
The documentation for this class was generated from the following files:

- [src/entity.hpp](#)
- [src/entity.cpp](#)

## 7.11 FloorTile Class Reference

```
#include <roomTile.hpp>
```

Inheritance diagram for FloorTile:



### Public Member Functions

- [FloorTile](#) (std::string texture, float x, float y)

### 7.11.1 Constructor & Destructor Documentation

#### 7.11.1.1 FloorTile()

```
FloorTile::FloorTile (  
    std::string texture,  
    float x,  
    float y ) [inline]
```

The documentation for this class was generated from the following file:

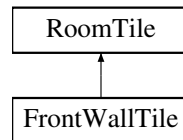
- [src/Dungeon/Tiles/roomTile.hpp](#)

## 7.12 FrontWallTile Class Reference

A wall that is not walkable but penetrable.

```
#include <roomTile.hpp>
```

Inheritance diagram for FrontWallTile:



### Public Member Functions

- [FrontWallTile](#) (std::string texture, float x, float y)

#### 7.12.1 Detailed Description

A wall that is not walkable but penetrable.

#### 7.12.2 Constructor & Destructor Documentation

##### 7.12.2.1 FrontWallTile()

```
FrontWallTile::FrontWallTile (  
    std::string texture,  
    float x,  
    float y ) [inline]
```

The documentation for this class was generated from the following file:

- src/Dungeon/Tiles/[roomTile.hpp](#)

## 7.13 Game Class Reference

```
#include <game.hpp>
```

## Public Member Functions

- [Game](#) ()
- [~Game](#) ()
- void [UpdateGame](#) ()  
*Updates the game instance/variables in the game loop.*
- void [RenderGame](#) ()  
*Renders the updated game variables in the game loop.*
- bool [Running](#) () const  
*Checks that the game is running aka window is open.*
- void [Events](#) ()  
*pulls events such as if the window is closed*

### 7.13.1 Constructor & Destructor Documentation

#### 7.13.1.1 Game()

```
Game::Game ( )
```

#### 7.13.1.2 ~Game()

```
Game::~Game ( )
```

### 7.13.2 Member Function Documentation

#### 7.13.2.1 Events()

```
void Game::Events ( )
```

pulls events such as if the window is closed

#### 7.13.2.2 RenderGame()

```
void Game::RenderGame ( )
```

Renders the updated game variables in the game loop.

### 7.13.2.3 Running()

```
bool Game::Running ( ) const
```

Checks that the game is running aka window is open.

#### Returns

true/false based if the window is open

### 7.13.2.4 UpdateGame()

```
void Game::UpdateGame ( )
```

Updates the game instance/variables in the game loop.

The documentation for this class was generated from the following files:

- [src/game.hpp](#)
- [src/game.cpp](#)

## 7.14 Gamebar Class Reference

```
#include <gamebar.hpp>
```

### Public Member Functions

- [Gamebar](#) ([PlayerPS](#) player)  
*Construct a new [Gamebar](#) object, meaning creates the gamebars for player health, attack cooldown and dash cooldown.*
- [Gamebar](#) ()
- void [Render](#) (sf::RenderTarget \*target)  
*renders the gamebars*
- void [Update](#) ()  
*keeps the gamebars updates when damage is lost and when attack and dash is used*
- void [RenderInventory](#) (sf::RenderTarget \*target)

### 7.14.1 Constructor & Destructor Documentation

#### 7.14.1.1 Gamebar() [1/2]

```
Gamebar::Gamebar (
    PlayerPS player )
```

Construct a new [Gamebar](#) object, meaning creates the gamebars for player health, attack cooldown and dash cooldown.

## Parameters

<i>player</i>	player which is tracked
---------------	-------------------------

**7.14.1.2 Gamebar()** [2/2]

```
Gamebar::Gamebar ( ) [inline]
```

**7.14.2 Member Function Documentation****7.14.2.1 Render()**

```
void Gamebar::Render (
    sf::RenderTarget * target )
```

renders the gamebars

## Parameters

<i>target</i>	window here gamebars should be drawn
---------------	--------------------------------------

**7.14.2.2 RenderInventory()**

```
void Gamebar::RenderInventory (
    sf::RenderTarget * target )
```

**7.14.2.3 Update()**

```
void Gamebar::Update ( )
```

keeps the gamebars updates when damage is lost and when attack and dash is used

The documentation for this class was generated from the following files:

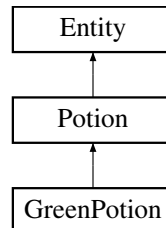
- [src/gamebar.hpp](#)
- [src/gamebar.cpp](#)

## 7.15 GreenPotion Class Reference

A green potion. Takes a position as parameter and the colour and healing effect is set automatically.

```
#include <HealthPotions.hpp>
```

Inheritance diagram for GreenPotion:



### Public Member Functions

- [GreenPotion](#) (sf::Vector2f pos)

### Additional Inherited Members

#### 7.15.1 Detailed Description

A green potion. Takes a position as parameter and the colour and healing effect is set automatically.

#### 7.15.2 Constructor & Destructor Documentation

##### 7.15.2.1 GreenPotion()

```
GreenPotion::GreenPotion (  
    sf::Vector2f pos ) [inline]
```

The documentation for this class was generated from the following file:

- src/Combat/Health/[HealthPotions.hpp](#)

## 7.16 ICollidable Class Reference

Unused interface for collisionsystem.

```
#include <ICollidable.hpp>
```



## Public Types

- enum [EntityType](#) { [character](#) , [projectile](#) , [tile](#) }

## Public Member Functions

- virtual sf::FloatRect [GetBoundingBox](#) ()=0
- virtual void [ProcessCollision](#) ([ICollidable](#) \*other)=0
- virtual [EntityType](#) [GetEntityType](#) ()=0
- virtual [~ICollidable](#) ()

### 7.16.1 Detailed Description

Unused interface for collisionsystem.

### 7.16.2 Member Enumeration Documentation

#### 7.16.2.1 EntityType

```
enum ICollidable::EntityType
```

Enumerator

character	
projectile	
tile	

### 7.16.3 Constructor & Destructor Documentation

#### 7.16.3.1 ~ICollidable()

```
virtual ICollidable::~ICollidable ( ) [inline], [virtual]
```

### 7.16.4 Member Function Documentation

#### 7.16.4.1 GetBoundingBox()

```
virtual sf::FloatRect ICollidable::GetBoundingBox ( ) [pure virtual]
```

#### 7.16.4.2 GetEntityType()

```
virtual EntityType ICollidable::GetEntityType ( ) [pure virtual]
```

#### 7.16.4.3 ProcessCollision()

```
virtual void ICollidable::ProcessCollision (
    ICollidable * other ) [pure virtual]
```

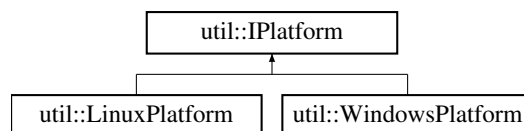
The documentation for this class was generated from the following file:

- [src/Interfaces/ICollidable.hpp](#)

## 7.17 util::IPlatform Struct Reference

```
#include <IPlatform.hpp>
```

Inheritance diagram for util::IPlatform:



### Public Member Functions

- virtual [~IPlatform](#) ()=default
- virtual void [setIcon](#) (const sf::WindowHandle &inHandle)=0
- virtual void [toggleFullscreen](#) (const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const bool in↵  
Windowed, const sf::Vector2u &inResolution)=0
- virtual int [getRefreshRate](#) (const sf::WindowHandle &inHandle)=0
- virtual float [getScreenScalingFactor](#) (const sf::WindowHandle &inHandle)=0

#### 7.17.1 Constructor & Destructor Documentation

### 7.17.1.1 ~IPlatform()

```
virtual util::IPlatform::~~IPlatform ( ) [virtual], [default]
```

## 7.17.2 Member Function Documentation

### 7.17.2.1 getRefreshRate()

```
virtual int util::IPlatform::getRefreshRate (
    const sf::WindowHandle & inHandle ) [pure virtual]
```

Implemented in [util::LinuxPlatform](#), and [util::WindowsPlatform](#).

### 7.17.2.2 getScreenScalingFactor()

```
virtual float util::IPlatform::getScreenScalingFactor (
    const sf::WindowHandle & inHandle ) [pure virtual]
```

Implemented in [util::LinuxPlatform](#), and [util::WindowsPlatform](#).

### 7.17.2.3 setIcon()

```
virtual void util::IPlatform::setIcon (
    const sf::WindowHandle & inHandle ) [pure virtual]
```

Implemented in [util::LinuxPlatform](#), and [util::WindowsPlatform](#).

### 7.17.2.4 toggleFullscreen()

```
virtual void util::IPlatform::toggleFullscreen (
    const sf::WindowHandle & inHandle,
    const sf::Uint32 inStyle,
    const bool inWindowed,
    const sf::Vector2u & inResolution ) [pure virtual]
```

Implemented in [util::LinuxPlatform](#), and [util::WindowsPlatform](#).

The documentation for this struct was generated from the following file:

- [src/Platform/IPlatform.hpp](#)

## 7.18 LevelUpInstance Class Reference

```
#include <LevelUpInstance.hpp>
```

### Public Member Functions

- [LevelUpInstance](#) ()  
*Construct a new Level Up Instance object, designed to only be used from the [LevelUpSystem](#). Initializes level as 1, xp as 0 and xpNeededForLevelUp as 20.*
- void [GainXP](#) (float amount)  
*Function to Add xp to this LevelUpInstace.*
- void [LevelUp](#) ()  
*Functio to Directly level up this [LevelUpInstance](#).*
- int [GetLevel](#) ()  
*Function to get the level of this [LevelUpInstance](#).*
- float [GetHPModifier](#) ()  
*Function to get HP modifier of this [LevelUpInstance](#).*

### 7.18.1 Constructor & Destructor Documentation

#### 7.18.1.1 LevelUpInstance()

```
LevelUpInstance::LevelUpInstance ( )
```

Construct a new Level Up Instance object, designed to only be used from the [LevelUpSystem](#). Initializes level as 1, xp as 0 and xpNeededForLevelUp as 20.

### 7.18.2 Member Function Documentation

#### 7.18.2.1 GainXP()

```
void LevelUpInstance::GainXP (
    float amount )
```

Function to Add xp to this LevelUpInstace.

#### Parameters

<i>amount</i>	Ammount to of XP add
---------------	----------------------

### 7.18.2.2 GetHPModifier()

```
float LevelUpInstance::GetHPModifier ( )
```

Function to get HP modifier of this [LevelUpInstance](#).

#### Returns

float HP modifier of this [LevelUpInstance](#)

### 7.18.2.3 GetLevel()

```
int LevelUpInstance::GetLevel ( ) [inline]
```

Function to get the level of this [LevelUpInstance](#).

#### Returns

int The level of this [LevelUpInstance](#)

### 7.18.2.4 LevelUp()

```
void LevelUpInstance::LevelUp ( )
```

Function to Directly level up this [LevelUpInstance](#).

The documentation for this class was generated from the following files:

- [src/Utility/LevelUpInstance.hpp](#)
- [src/Utility/LevelUpInstance.cpp](#)

## 7.19 LevelUpSystem Class Reference

```
#include <LevelUpSystem.hpp>
```

### Static Public Member Functions

- static void [AddCharacter](#) ([Character](#) \*character)  
*Function for adding a character to the [LevelUpSystem](#).*
- static void [GainXP](#) ([Character](#) \*character, float amount)  
*Function to add XP to a character that is in the [LevelUpSystem](#).*
- static void [LevelUp](#) ([Character](#) \*character)  
*Function to directly level up a character. Sets character XP to 0.*
- static int [GetLevel](#) ([Character](#) \*character)  
*Function to get the level of a character in the [LevelUpSystem](#).*
- static float [GetHPModifier](#) ([Character](#) \*character)  
*Function to get a HP modifier that is calculated from the level of the character. Used to buff character when leveling up.*

## Static Public Attributes

- static std::unordered\_map< [Character](#) \*, [LevelUpInstance](#) > [characterLevelMap](#)

## 7.19.1 Member Function Documentation

### 7.19.1.1 AddCharacter()

```
void LevelUpSystem::AddCharacter (  
    Character * character ) [static]
```

Function for adding a character to the [LevelUpSystem](#).

#### Parameters

<i>character</i>	<a href="#">Character</a> to add to the <a href="#">LevelUpSystem</a>
------------------	---

### 7.19.1.2 GainXP()

```
void LevelUpSystem::GainXP (  
    Character * character,  
    float amount ) [static]
```

Function to add XP to a character that is in the [LevelUpSystem](#).

#### Parameters

<i>character</i>	<a href="#">Character</a> to add XP to
<i>amount</i>	Amount of XP to add to the character

### 7.19.1.3 GetHPModifier()

```
float LevelUpSystem::GetHPModifier (  
    Character * character ) [static]
```

Function to get a HP modifier that is calculated from the level of the character. Used to buff character when leveling up.

#### Parameters

<i>character</i>	<a href="#">Character</a> to get HP modifier of
------------------	---

**Returns**

float The HP modifier of the character

**7.19.1.4 GetLevel()**

```
int LevelUpSystem::GetLevel (
    Character * character ) [static]
```

Function to get the level of a character in the [LevelUpSystem](#).

**Parameters**

<i>character</i>	<a href="#">Character</a> to get level of
------------------	---

**Returns**

int The level of the character

**7.19.1.5 LevelUp()**

```
void LevelUpSystem::LevelUp (
    Character * character ) [static]
```

Function to directly level up a character. Sets character XP to 0.

**Parameters**

<i>character</i>	<a href="#">Character</a> to level up
------------------	---------------------------------------

**7.19.2 Member Data Documentation****7.19.2.1 characterLevelMap**

```
std::unordered_map< Character *, LevelUpInstance > LevelUpSystem::characterLevelMap [static]
```

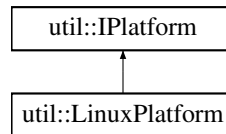
The documentation for this class was generated from the following files:

- [src/Utility/LevelUpSystem.hpp](#)
- [src/Utility/LevelUpSystem.cpp](#)

## 7.20 util::LinuxPlatform Struct Reference

```
#include <LinuxPlatform.hpp>
```

Inheritance diagram for util::LinuxPlatform:



### Public Member Functions

- [LinuxPlatform](#) ()
- void [setIcon](#) (const sf::WindowHandle &inHandle) final
- void [toggleFullscreen](#) (const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const bool inWindowed, const sf::Vector2u &inResolution) final
- float [getScreenScalingFactor](#) (const sf::WindowHandle &inHandle) final
- int [getRefreshRate](#) (const sf::WindowHandle &inHandle) final

### 7.20.1 Constructor & Destructor Documentation

#### 7.20.1.1 LinuxPlatform()

```
util::LinuxPlatform::LinuxPlatform ( )
```

### 7.20.2 Member Function Documentation

#### 7.20.2.1 getRefreshRate()

```
int util::LinuxPlatform::getRefreshRate (
    const sf::WindowHandle & inHandle ) [final], [virtual]
```

Implements [util::IPlatform](#).

#### 7.20.2.2 getScreenScalingFactor()

```
float util::LinuxPlatform::getScreenScalingFactor (
    const sf::WindowHandle & inHandle ) [final], [virtual]
```

Implements [util::IPlatform](#).



### 7.20.2.3 setIcon()

```
void util::LinuxPlatform::setIcon (
    const sf::WindowHandle & inHandle ) [final], [virtual]
```

Implements [util::IPlatform](#).

### 7.20.2.4 toggleFullscreen()

```
void util::LinuxPlatform::toggleFullscreen (
    const sf::WindowHandle & inHandle,
    const sf::Uint32 inStyle,
    const bool inWindowed,
    const sf::Vector2u & inResolution ) [final], [virtual]
```

Implements [util::IPlatform](#).

The documentation for this struct was generated from the following file:

- [src/Platform/Unix/LinuxPlatform.hpp](#)

## 7.21 Map Class Reference

```
#include <map.hpp>
```

### Public Member Functions

- [Map](#) (sf::Vector2u sizeOfRooms, int noRooms, [PlayerPS](#) player)  
*Initializes variables for new room and calls [CreateDungeon\(\)](#)*
- [~Map](#) ()
- void [RenderCurrentRoom](#) (sf::RenderTarget \*target)  
*Renders the current room to the given target.*
- void [CreateDungeon](#) (int noRooms)  
*Create the actual dungeon.*
- void [MovePlayer](#) ([Direction](#) dir)  
*Moves the player to the room on the map in the given direction.*
- [RoomInstance](#) \* [GetRoomInDir](#) ([Direction](#) direction)  
*Get the Room in the argument direction.*
- sf::Vector2i [DirToVec](#) ([Direction](#) direction)  
*Converts a Direction to a unit vector and returns it.*
- [RoomInstance](#) \* [GetCurrentRoom](#) ()  
*Get the room the player is in on the map.*
- [RoomInstance](#) \* [GetSpawnRoom](#) ()  
*Get the SpawnRoom.*
- void [ResetMap](#) ()
- bool [IsBossRoomCleared](#) ()  
*Checks Whether the bossroom has been cleared.*

## 7.21.1 Constructor & Destructor Documentation

### 7.21.1.1 Map()

```
Map::Map (
    sf::Vector2u sizeOfRooms,
    int noRooms,
    PlayerPS player )
```

Initializes variables for new room and calls [CreateDungeon\(\)](#)

#### Parameters

<i>sizeOfRooms</i>	the size we want our rooms to be
<i>noRooms</i>	The number of rooms we want the dungeon to consist of

### 7.21.1.2 ~Map()

```
Map::~Map ( )
```

## 7.21.2 Member Function Documentation

### 7.21.2.1 CreateDungeon()

```
void Map::CreateDungeon (
    int noRooms )
```

Create the actual dungeon.

#### Parameters

<i>noRooms</i>	number of rooms on in the dungeon
----------------	-----------------------------------

### 7.21.2.2 DirToVec()

```
sf::Vector2i Map::DirToVec (
    Direction direction )
```

Converts a Direction to a unit vector and returns it.

## Parameters

<i>direction</i>	the direction
------------------	---------------

## Returns

sf::Vector2i the converted vector

### 7.21.2.3 GetCurrentRoom()

```
RoomInstance * Map::GetCurrentRoom ( )
```

Get the room the player is in on the map.

## Returns

RoomInstance\* The room

### 7.21.2.4 GetRoomInDir()

```
RoomInstance * Map::GetRoomInDir (
    Direction direction )
```

Get the Room in the argument direction.

## Parameters

<i>direction</i>	The direction
------------------	---------------

## Returns

returns room, or nullptr if not found

### 7.21.2.5 GetSpawnRoom()

```
RoomInstance * Map::GetSpawnRoom ( )
```

Get the SpawnRoom.

## Returns

RoomInstance\* The SpawnRoom

#### 7.21.2.6 IsBossRoomCleared()

```
bool Map::IsBossRoomCleared ( )
```

Checks Whether the bossroom has been cleared.

##### Returns

true  
false

#### 7.21.2.7 MovePlayer()

```
void Map::MovePlayer (
    Direction dir )
```

Moves the player to the room on the map in the given direction.

##### Parameters

<i>dir</i>	Direction to go
------------	-----------------

#### 7.21.2.8 RenderCurrentRoom()

```
void Map::RenderCurrentRoom (
    sf::RenderTarget * target )
```

Renders the current room to the given target.

#### 7.21.2.9 ResetMap()

```
void Map::ResetMap ( )
```

The documentation for this class was generated from the following files:

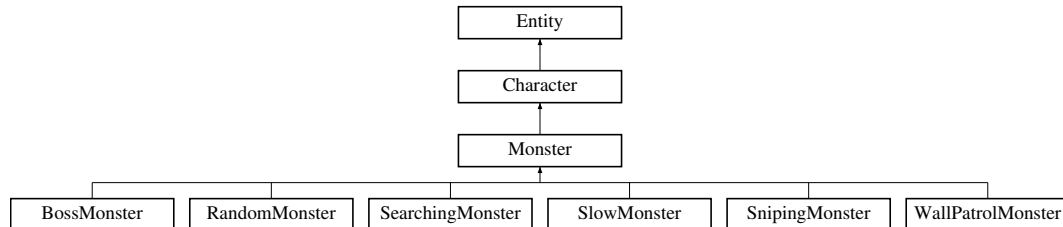
- [src/Dungeon/map.hpp](#)
- [src/Dungeon/map.cpp](#)

## 7.22 Monster Class Reference

[Monster](#) class, which all our other monsters inherit.

```
#include <monster.hpp>
```

Inheritance diagram for Monster:



### Public Member Functions

- virtual [~Monster](#) ()
- [Player & GetPlayer](#) () const
- virtual std::list< [ProjectileUP](#) > [Attack](#) ()=0
- virtual void [Update](#) (float)
- virtual bool [Move](#) (float dt)=0
- virtual void [Render](#) (sf::RenderTarget \*target)
- void [initVariables](#) ()
- void [SetTarget](#) ([PlayerPS](#) target)
- [Potion](#) \* [ReturnPotion](#) ()

*determines if monster is going to drop a potion and what kind of potion.*

### Protected Member Functions

- [Monster](#) ([PlayerPS](#) player, sf::Vector2f pos, const std::string &spriteFile)
- [Monster](#) ([PlayerPS](#) player, float xPos, float yPos, const std::string &spriteFile)
- float [getDistanceToPlayer](#) ()
- bool [inRangeOfPlayer](#) ()
- bool [moveTowardsPlayer](#) (float dt)
- void [clampPosToRoom](#) ()

*Checks if player is in attack range of monster character based on weapon range.*

*clamps the monsters into a room so they can go into corridors*

### Protected Attributes

- std::shared\_ptr< [Player](#) > [player\\_](#)
- sf::RectangleShape [healthbar\\_](#)
- float [staticDamage\\_](#) = 5.0f
- bool [movedLastTick\\_](#)

## Additional Inherited Members

### 7.22.1 Detailed Description

[Monster](#) class, which all our other monsters inherit.

### 7.22.2 Constructor & Destructor Documentation

#### 7.22.2.1 ~Monster()

```
Monster::~Monster ( ) [virtual]
```

#### 7.22.2.2 Monster() [1/2]

```
Monster::Monster (
    PlayerPS player,
    sf::Vector2f pos,
    const std::string & spriteFile ) [protected]
```

#### 7.22.2.3 Monster() [2/2]

```
Monster::Monster (
    PlayerPS player,
    float xPos,
    float yPos,
    const std::string & spriteFile ) [protected]
```

### 7.22.3 Member Function Documentation

#### 7.22.3.1 Attack()

```
virtual std::list< ProjectileUP > Monster::Attack ( ) [pure virtual]
```

Implemented in [BossMonster](#), [RandomMonster](#), [SearchingMonster](#), [SlowMonster](#), [SnipingMonster](#), and [WallPatrolMonster](#).

### 7.22.3.2 clampPosToRoom()

```
void Monster::clampPosToRoom ( ) [protected]
```

clamps the monsters into a room so they can go into corridors

### 7.22.3.3 getDistanceToPlayer()

```
float Monster::getDistanceToPlayer ( ) [protected]
```

### 7.22.3.4 GetPlayer()

```
Player & Monster::GetPlayer ( ) const
```

### 7.22.3.5 initVariables()

```
void Monster::initVariables ( )
```

### 7.22.3.6 inRangeOfPlayer()

```
bool Monster::inRangeOfPlayer ( ) [protected]
```

Checks if player is in attack range of monster character based on weapon range.

#### Returns

true if it is  
false if not

### 7.22.3.7 Move()

```
virtual bool Monster::Move (
    float dt ) [pure virtual]
```

Reimplemented from [Character](#).

Implemented in [BossMonster](#), [RandomMonster](#), [SearchingMonster](#), [SlowMonster](#), [SnipingMonster](#), and [WallPatrolMonster](#).

### 7.22.3.8 moveTowardsPlayer()

```
bool Monster::moveTowardsPlayer (
    float dt ) [protected]
```

### 7.22.3.9 Render()

```
void Monster::Render (
    sf::RenderTarget * target ) [virtual]
```

Reimplemented from [Entity](#).

### 7.22.3.10 ReturnPotion()

```
Potion * Monster::ReturnPotion ( )
```

determines if monster is going to drop a potion and what kind of potion.

#### Returns

Potion\*

### 7.22.3.11 SetTarget()

```
void Monster::SetTarget (
    PlayerPS target )
```

### 7.22.3.12 Update()

```
void Monster::Update (
    float dt ) [virtual]
```

Implements [Character](#).

## 7.22.4 Member Data Documentation



**7.22.4.1 healthbar\_**

```
sf::RectangleShape Monster::healthbar_ [protected]
```

**7.22.4.2 movedLastTick\_**

```
bool Monster::movedLastTick_ [protected]
```

**7.22.4.3 player\_**

```
std::shared_ptr<Player> Monster::player_ [protected]
```

**7.22.4.4 staticDamage\_**

```
float Monster::staticDamage_ = 5.0f [protected]
```

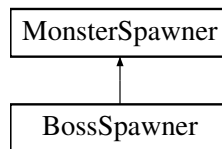
The documentation for this class was generated from the following files:

- src/Actors/Monsters/[monster.hpp](#)
- src/Actors/Monsters/[monster.cpp](#)

**7.23 MonsterSpawner Class Reference**

```
#include <MonsterSpawner.hpp>
```

Inheritance diagram for MonsterSpawner:

**Public Member Functions**

- [MonsterSpawner](#) (uint monsterAmount)
- [MonsterSpawner](#) ()
- virtual [~MonsterSpawner](#) ()
- virtual [MonsterSP SpawnMonster](#) (sf::Vector2u roomSize, [PlayerPS](#) target) const
- void [SetMonsterAmount](#) (uint amount)
- [uint GetMonsterAmount](#) () const

## Protected Member Functions

- [MonsterSP getRandomMonster](#) ([PlayerPS](#) target) const

## Protected Attributes

- [uint monsterCount\\_](#)
- [uint monsterTypeCount\\_](#) = 5

## 7.23.1 Constructor & Destructor Documentation

### 7.23.1.1 MonsterSpawner() [1/2]

```
MonsterSpawner::MonsterSpawner (
    uint monsterAmount ) [inline]
```

### 7.23.1.2 MonsterSpawner() [2/2]

```
MonsterSpawner::MonsterSpawner ( ) [inline]
```

### 7.23.1.3 ~MonsterSpawner()

```
virtual MonsterSpawner::~~MonsterSpawner ( ) [inline], [virtual]
```

## 7.23.2 Member Function Documentation

### 7.23.2.1 GetMonsterAmount()

```
uint MonsterSpawner::GetMonsterAmount ( ) const
```

### 7.23.2.2 getRandomMonster()

```
MonsterSP MonsterSpawner::getRandomMonster (
    PlayerPS target ) const [protected]
```

### 7.23.2.3 SetMonsterAmount()

```
void MonsterSpawner::SetMonsterAmount (
    uint amount )
```

### 7.23.2.4 SpawnMonster()

```
MonsterSP MonsterSpawner::SpawnMonster (
    sf::Vector2u roomSize,
    PlayerPS target ) const [virtual]
```

## 7.23.3 Member Data Documentation

### 7.23.3.1 monsterCount\_

```
uint MonsterSpawner::monsterCount_ [protected]
```

### 7.23.3.2 monsterTypeCount\_

```
uint MonsterSpawner::monsterTypeCount_ = 5 [protected]
```

The documentation for this class was generated from the following files:

- src/Actors/Monsters/MonsterSpawner/[MonsterSpawner.hpp](#)
- src/Actors/Monsters/MonsterSpawner/[MonsterSpawner.cpp](#)

## 7.24 Collision::OrientedBoundingBox Class Reference

### Public Member Functions

- [OrientedBoundingBox](#) (const sf::Sprite &Object)
- void [ProjectOntoAxis](#) (const sf::Vector2f &Axis, float &Min, float &Max)

### Public Attributes

- sf::Vector2f [Points](#) [4]

## 7.24.1 Constructor & Destructor Documentation

### 7.24.1.1 OrientedBoundingBox()

```
Collision::OrientedBoundingBox::OrientedBoundingBox (
    const sf::Sprite & Object ) [inline]
```

## 7.24.2 Member Function Documentation

### 7.24.2.1 ProjectOntoAxis()

```
void Collision::OrientedBoundingBox::ProjectOntoAxis (
    const sf::Vector2f & Axis,
    float & Min,
    float & Max ) [inline]
```

## 7.24.3 Member Data Documentation

### 7.24.3.1 Points

```
sf::Vector2f Collision::OrientedBoundingBox::Points[4]
```

The documentation for this class was generated from the following file:

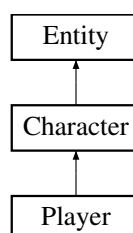
- [src/Utility/Collision.cpp](#)

## 7.25 Player Class Reference

[Player](#) class, our controlled character.

```
#include <player.hpp>
```

Inheritance diagram for Player:



## Public Member Functions

- [Player](#) ()
- [~Player](#) ()
- virtual void [Update](#) (float dt)
- void [Dash](#) ()
  - Handles the dashing functions.*
- std::list< [ProjectileUP](#) > [Attack](#) (sf::Vector2f aimPos)
  - handles the attacking of player character*
- void [initVariables](#) ()
- void [ResetDashCooldown](#) ()
- float [GetDashCooldownLeft](#) () const
- float [GetDashCooldownLength](#) () const
- void [AddPotion](#) ([Potion](#) \*potion)
  - Adds a potion to be used later.*
- void [UsePotion](#) (const std::string &colour)
  - Get the colour of the potion player is going to use and runs the functions to use it.*
- std::vector< [Potion](#) \* > [GetInventory](#) () const
  - gets inventory of*
- void [ClearInventory](#) ()

## Public Attributes

- bool [CanDash](#)
- bool [IsDashing](#)

## Additional Inherited Members

### 7.25.1 Detailed Description

[Player](#) class, our controlled character.

### 7.25.2 Constructor & Destructor Documentation

#### 7.25.2.1 [Player\(\)](#)

```
Player::Player ( )
```

#### 7.25.2.2 [~Player\(\)](#)

```
Player::~~Player ( ) [inline]
```

### 7.25.3 Member Function Documentation

#### 7.25.3.1 AddPotion()

```
void Player::AddPotion (  
    Potion * potion )
```

Adds a potion to be used later.

## Parameters

<i>potion</i>	potion to be added to inventory
---------------	---------------------------------

**7.25.3.2 Attack()**

```
std::list< ProjectileUP > Player::Attack (
    sf::Vector2f aimPos )
```

handles the attacking of player character

## Parameters

<i>aimPos</i>	position of where mouse is aiming
---------------	-----------------------------------

## Returns

```
std::list<ProjectileUP>
```

**7.25.3.3 ClearInventory()**

```
void Player::ClearInventory ( )
```

**7.25.3.4 Dash()**

```
void Player::Dash ( )
```

Handles the dashing functions.

**7.25.3.5 GetDashCooldownLeft()**

```
float Player::GetDashCooldownLeft ( ) const [inline]
```

**7.25.3.6 GetDashCooldownLength()**

```
float Player::GetDashCooldownLength ( ) const [inline]
```

### 7.25.3.7 GetInventory()

```
std::vector< Potion * > Player::GetInventory ( ) const
```

gets inventory of

#### Returns

std::vector<Potion\*>

### 7.25.3.8 initVariables()

```
void Player::initVariables ( )
```

### 7.25.3.9 ResetDashCooldown()

```
void Player::ResetDashCooldown ( )
```

### 7.25.3.10 Update()

```
void Player::Update (
    float dt ) [virtual]
```

Implements [Character](#).

### 7.25.3.11 UsePotion()

```
void Player::UsePotion (
    const std::string & colour )
```

Get the colour of the potion player is going to use and runs the functions to use it.

#### Parameters

<i>colour</i>	colour of potion
---------------	------------------



## 7.25.4 Member Data Documentation

### 7.25.4.1 CanDash

```
bool Player::CanDash
```

### 7.25.4.2 IsDashing

```
bool Player::IsDashing
```

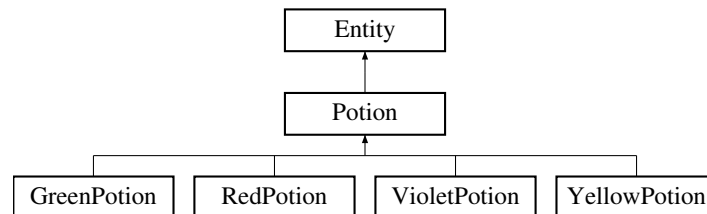
The documentation for this class was generated from the following files:

- [src/Actors/player.hpp](#)
- [src/Actors/player.cpp](#)

## 7.26 Potion Class Reference

```
#include <Potion.hpp>
```

Inheritance diagram for Potion:



### Public Member Functions

- [Potion](#) (const std::string &spritefile, sf::Vector2f pos, int healthIncrease, const std::string &colour)
- virtual [~Potion](#) ()
- const std::string & [GetColour](#) () const  
*Get the Colour of the potion.*
- int [GetHealthIncrease](#) () const  
*Get the Healing effect of the potion.*

### Protected Attributes

- int [healthIncrease\\_](#)
- std::string [colour\\_](#)

## Additional Inherited Members

### 7.26.1 Constructor & Destructor Documentation

#### 7.26.1.1 Potion()

```
Potion::Potion (
    const std::string & spritefile,
    sf::Vector2f pos,
    int healthIncrease,
    const std::string & colour )
```

#### 7.26.1.2 ~Potion()

```
virtual Potion::~~Potion ( ) [inline], [virtual]
```

### 7.26.2 Member Function Documentation

#### 7.26.2.1 GetColour()

```
const std::string & Potion::GetColour ( ) const
```

Get the Colour of the potion.

##### Returns

const refrence to a string with the color

#### 7.26.2.2 GetHealthIncrease()

```
int Potion::GetHealthIncrease ( ) const
```

Get the Healing effect of the potion.

##### Returns

int

### 7.26.3 Member Data Documentation

#### 7.26.3.1 colour\_

```
std::string Potion::colour_ [protected]
```

#### 7.26.3.2 healthIncrease\_

```
int Potion::healthIncrease_ [protected]
```

The documentation for this class was generated from the following files:

- [src/Combat/Health/Potion.hpp](#)
- [src/Combat/Health/Potion.cpp](#)

## 7.27 PowerUp Class Reference

```
#include <PowerUp.hpp>
```

### Public Member Functions

- [PowerUp\(\)](#)
- [~PowerUp\(\)](#)

### 7.27.1 Constructor & Destructor Documentation

#### 7.27.1.1 PowerUp()

```
PowerUp::PowerUp ( ) [inline]
```

#### 7.27.1.2 ~PowerUp()

```
PowerUp::~PowerUp ( ) [inline]
```

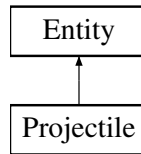
The documentation for this class was generated from the following file:

- [src/Combat/PowerUps/PowerUp.hpp](#)

## 7.28 Projectile Class Reference

```
#include <Projectile.hpp>
```

Inheritance diagram for Projectile:



### Public Types

- enum [Type](#) { [PlayerProjectile](#) , [EnemyProjectile](#) }

### Public Member Functions

- [Projectile](#) (sf::Sprite &sprite, sf::Vector2f startPos, bool penetratesObjects=false)
- bool [HasHit](#) ([Character](#) \*c)  
*Function to check if this projectile has already hit a specific character.*
- void [Hit](#) ([Character](#) \*c)  
*Function to add character c to the characters this projectile has hit so it cannot hit it again.*
- int [GetDamage](#) ()  
*Get the Damage that this projectile will make when colliding.*
- [Projectile::Type](#) [GetType](#) ()  
*Get the Type of this projectile.*
- sf::Vector2f [GetDirection](#) ()  
*Get the Direction that this [Projectile](#) is travelling in.*
- sf::Vector2f [GetStartPosition](#) ()  
*Get the position that this [Projectile](#) was spawned at.*
- float [GetTimeExisted](#) ()  
*Get the time in seconds that this [Projectile](#) has existed.*
- float [GetTimeLifeSpan](#) ()  
*Get the time in seconds that this [Projectile](#) will exist before being deleted.*
- float [GetProjectileSpeed](#) ()  
*Get the speed that this [Projectile](#) is travelling at.*
- float [GetDistanceLifeSpan](#) ()  
*Get the distance that this [Projectile](#) will travel before being deleted.*
- bool [Penetrates](#) ()  
*Check if this [Projectile](#) can penetrate characters or not. Meaning if it can go through characters.*
- bool [IsAlive](#) ()  
*Checks if this [Projectile](#) should be deleted or not.*
- void [SetType](#) ([Projectile::Type](#) type)  
*Set the Type of this projectile.*
- void [SetDirection](#) (sf::Vector2f direction)  
*Set the Direction of this projectile.*
- void [SetDamage](#) (int damage)  
*Set the Damage of this projectile.*

- void [SetProjectileSpeed](#) (float projectileSpeed)  
*Set the Speed of this projectile.*
- void [SetTimeLifeSpan](#) (float timeLifeSpan)  
*Set the timeLifeSpan of this projectile.*
- void [SetDistanceLifeSpan](#) (float distanceLifeSpan)  
*Set the distanceLifeSpan of this projectile.*
- void [SetSprite](#) (sf::Sprite sprite)  
*Set the Sprite of this [Projectile](#).*
- void [Kill](#) ()  
*Function to make the game delete this [Projectile](#).*
- void [Update](#) (float dt)  
*Function to update this projectile.*

## Additional Inherited Members

### 7.28.1 Member Enumeration Documentation

#### 7.28.1.1 Type

```
enum Projectile::Type
```

Enumerator

PlayerProjectile	
EnemyProjectile	

### 7.28.2 Constructor & Destructor Documentation

#### 7.28.2.1 Projectile()

```
Projectile::Projectile (  
    sf::Sprite & sprite,  
    sf::Vector2f startPos,  
    bool penetratesObjects = false )
```

### 7.28.3 Member Function Documentation

### 7.28.3.1 GetDamage()

```
int Projectile::GetDamage ( ) [inline]
```

Get the Damage that this projectile will make when colliding.

#### Returns

int damage of this projectile

### 7.28.3.2 GetDirection()

```
sf::Vector2f Projectile::GetDirection ( ) [inline]
```

Get the Direction that this [Projectile](#) is travelling in.

#### Returns

sf::Vector2f direction of this projectile

### 7.28.3.3 GetDistanceLifeSpan()

```
float Projectile::GetDistanceLifeSpan ( ) [inline]
```

Get the distance that this [Projectile](#) will travel before being deleted.

#### Returns

float distanceLifeSpan of this [Projectile](#)

### 7.28.3.4 GetProjectileSpeed()

```
float Projectile::GetProjectileSpeed ( ) [inline]
```

Get the speed that this [Projectile](#) is travelling at.

#### Returns

float speed of this [Projectile](#)

### 7.28.3.5 GetStartPosition()

```
sf::Vector2f Projectile::GetStartPosition ( ) [inline]
```

Get the position that this [Projectile](#) was spawned at.

#### Returns

sf::Vector2f start position of this [Projectile](#)

### 7.28.3.6 GetTimeExisted()

```
float Projectile::GetTimeExisted ( ) [inline]
```

Get the time in seconds that this [Projectile](#) has existed.

#### Returns

float timeExisted of this [Projectile](#) in seconds

### 7.28.3.7 GetTimeLifeSpan()

```
float Projectile::GetTimeLifeSpan ( ) [inline]
```

Get the time in seconds that this [Projectile](#) will exist before being deleted.

#### Returns

float timeLifeSpan of this [Projectile](#) in seconds

### 7.28.3.8 GetType()

```
Projectile::Type Projectile::GetType ( ) [inline]
```

Get the Type of this projectile.

#### Returns

[Projectile::Type](#) the type of this projectile

### 7.28.3.9 HasHit()

```
bool Projectile::HasHit (
    Character * c )
```

Function to check if this projectile has already hit a specific character.

**Parameters**

c	character to check if this projectile has hit
---	---

**Returns**

true if projectile has hit character c  
false if projectile has not hit character c

**7.28.3.10 Hit()**

```
void Projectile::Hit (
    Character * c )
```

Function to add character c to the characters this projectile has hit so it cannot hit it again.

**Parameters**

c	character to add
---	------------------

**7.28.3.11 IsAlive()**

```
bool Projectile::IsAlive ( ) [inline]
```

Checks if this [Projectile](#) should be deleted or not.

**Returns**

true Meaning that this [Projectile](#) should not be deleted  
false Meaning that this [Projectile](#) should be deleted

**7.28.3.12 Kill()**

```
void Projectile::Kill ( ) [inline]
```

Function to make the game delete this [Projectile](#).



### 7.28.3.13 Penetrates()

```
bool Projectile::Penetrates ( ) [inline]
```

Check if this [Projectile](#) can penetrate characters or not. Meaning if it can go through characters.

#### Returns

- true Meaning that this [Projectile](#) can penetrate characters
- false Meaning that this [Projectile](#) can not penetrate characters

### 7.28.3.14 SetDamage()

```
void Projectile::SetDamage (
    int damage )
```

Set the Damage of this projectile.

#### Parameters

<i>damage</i>	the new Damage of this projectile
---------------	-----------------------------------

### 7.28.3.15 SetDirection()

```
void Projectile::SetDirection (
    sf::Vector2f direction )
```

Set the Direction of this projectile.

#### Parameters

<i>direction</i>	the new Direction of this projectile
------------------	--------------------------------------

### 7.28.3.16 SetDistanceLifeSpan()

```
void Projectile::SetDistanceLifeSpan (
    float distanceLifeSpan )
```

Set the distanceLifeSpan of this projectile.

## Parameters

<i>distanceLifeSpan</i>	the new distanceLifeSpan of this projectile
-------------------------	---

**7.28.3.17 SetProjectileSpeed()**

```
void Projectile::SetProjectileSpeed (
    float projectileSpeed )
```

Set the Speed of this projectile.

## Parameters

<i>projectileSpeed</i>	the new Speed of this projectile
------------------------	----------------------------------

**7.28.3.18 SetSprite()**

```
void Projectile::SetSprite (
    sf::Sprite sprite ) [inline]
```

Set the Sprite of this [Projectile](#).

## Parameters

<i>sprite</i>	the new Sprite of this projectile
---------------	-----------------------------------

**7.28.3.19 SetTimeLifeSpan()**

```
void Projectile::SetTimeLifeSpan (
    float timeLifeSpan )
```

Set the timeLifeSpan of this projectile.

## Parameters

<i>timeLifeSpan</i>	the new timeLifeSpan of this projectile
---------------------	---

### 7.28.3.20 SetType()

```
void Projectile::SetType (
    Projectile::Type type )
```

Set the Type of this projectile.

#### Parameters

<i>type</i>	the new type of this projectile
-------------	---------------------------------

### 7.28.3.21 Update()

```
void Projectile::Update (
    float dt )
```

Function to update this projectile.

#### Parameters

<i>dt</i>	deltatime to compensate framerate
-----------	-----------------------------------

The documentation for this class was generated from the following files:

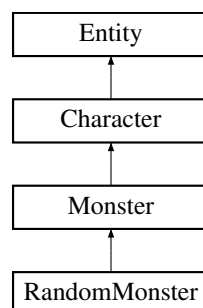
- src/Combat/[Projectile.hpp](#)
- src/Combat/[projectile.cpp](#)

## 7.29 RandomMonster Class Reference

[RandomMonster](#) is a monster that moves randomly around and shoots projectiles towards the player.

```
#include <RandomMonster.hpp>
```

Inheritance diagram for RandomMonster:



## Public Member Functions

- [RandomMonster](#) ([PlayerPS](#) player, float xPos, float yPos)
- [RandomMonster](#) ([PlayerPS](#) player, sf::Vector2f pos)
- [~RandomMonster](#) ()
- virtual std::list< [ProjectileUP](#) > [Attack](#) ()
- virtual bool [Move](#) (float dt)
- void [initVariables](#) ()

## Additional Inherited Members

### 7.29.1 Detailed Description

[RandomMonster](#) is a monster that moves randomly around and shoots projectiles towards the player.

### 7.29.2 Constructor & Destructor Documentation

#### 7.29.2.1 [RandomMonster\(\)](#) [1/2]

```
RandomMonster::RandomMonster (
    PlayerPS player,
    float xPos,
    float yPos )
```

#### 7.29.2.2 [RandomMonster\(\)](#) [2/2]

```
RandomMonster::RandomMonster (
    PlayerPS player,
    sf::Vector2f pos )
```

#### 7.29.2.3 [~RandomMonster\(\)](#)

```
RandomMonster::~~RandomMonster ( )
```

### 7.29.3 Member Function Documentation

### 7.29.3.1 Attack()

```
std::list< ProjectileUP > RandomMonster::Attack ( ) [virtual]
```

Implements [Monster](#).

### 7.29.3.2 initVariables()

```
void RandomMonster::initVariables ( )
```

### 7.29.3.3 Move()

```
bool RandomMonster::Move (
    float dt ) [virtual]
```

Implements [Monster](#).

The documentation for this class was generated from the following files:

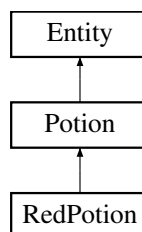
- [src/Actors/Monsters/RandomMonster.hpp](#)
- [src/Actors/Monsters/RandomMonster.cpp](#)

## 7.30 RedPotion Class Reference

A red potion. Takes a position as parameter and the colour and healing effect is set automatically.

```
#include <HealthPotions.hpp>
```

Inheritance diagram for RedPotion:



### Public Member Functions

- [RedPotion](#) (sf::Vector2f pos)

## Additional Inherited Members

### 7.30.1 Detailed Description

A red potion. Takes a position as parameter and the colour and healing effect is set automatically.

### 7.30.2 Constructor & Destructor Documentation

#### 7.30.2.1 RedPotion()

```
RedPotion::RedPotion (
    sf::Vector2f pos ) [inline]
```

The documentation for this class was generated from the following file:

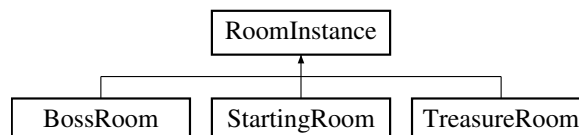
- src/Combat/Health/[HealthPotions.hpp](#)

## 7.31 RoomInstance Class Reference

Class representing a room of a dungeon, usually includes a monsterspawner.

```
#include <roomInstance.hpp>
```

Inheritance diagram for RoomInstance:



## Public Member Functions

- [RoomInstance](#) (sf::Vector2u window\_size, sf::Vector2i coords)  
*Construct a new Room Instance object.*
- [RoomInstance](#) (sf::Vector2u window\_size, sf::Vector2i coords, [MonsterSpawner](#) \*spawner)
- [RoomInstance](#) ()=default
- virtual [~RoomInstance](#) ()
- void [Render](#) (sf::RenderTarget \*target)  
*Renders the room.*
- std::vector< [RoomTile](#) \* > [getRoomTilesAt](#) (sf::FloatRect bounds)
- bool [positionIsWalkable](#) (sf::FloatRect bounds)
- bool [positionIsPenetratable](#) (sf::FloatRect bounds)
- std::vector< std::vector< [RoomTile](#) \* > > [getTiles](#) () const
- void [CreateExit](#) ([Direction](#) direction)

- Takes down a 2-wide in wall (replaces walls with floortiles) to be able to walk between two rooms.*

  - void [Connect](#) ([Direction](#) dir, [RoomInstance](#) \*room)

*Connects two rooms to each other.*
- [RoomInstance](#) \* [GetRoomInDir](#) ([Direction](#) dir)

*Returns the room found in the wanted direction, nullptr if it is not found.*
- [sf::Vector2i](#) [GetCoords](#) ()

*Get the coordinates of this room on the map.*
- void [renderSpriteBackground](#) ()

*Renders the rooms background to be a static backdrop, a very expensive operation.*
- [sf::Vector2u](#) [GetEntranceInDirection](#) ([Direction](#) direction)
- virtual void [Enter](#) ([PlayerPS](#) player, [Direction](#) direction)

*Function called when entering a room, needs player as parameter to perform some calculations about room difficulty.*
- void [Exit](#) ()

*Function called when exiting a room modifies cleared\_ and visited\_ booleans.*
- [std::vector](#)< [MonsterSP](#) > & [GetMonsters](#) ()
- void [deleteMonster](#) ([MonsterSP](#) m)
- [Direction](#) [RemoveRandomDirection](#) ()

*Function used for Generating the dungeon map. Removes and returns a random Direction from the directionsLeft\_ vector.*
- void [RemoveDirection](#) ([Direction](#) dir)

*Function used for Generating the dungeon map. Removes the Direction dir from the directionsLeft\_ vector.*
- bool [HasDirectionsLeft](#) ()

*Function used for Generating the dungeon map. Checks if this roomInstance has any direction that does not already contain an other roomInstance.*
- bool [IsCleared](#) ()

*Function the check if the roomInstance is cleared of Monsters.*
- bool [IsVisisted](#) ()
- bool [monsterCleared](#) ()
- void [AddPotion](#) ([Potion](#) \*potion)
- [std::vector](#)< [Potion](#) \* > & [GetPotions](#) ()

## Protected Member Functions

- virtual void [setTiles](#) ()

*Helper, Sets all the roomtiles of the roominstance.*

## Protected Attributes

- [sf::Vector2u](#) [roomSize\\_](#)
- [sf::Vector2i](#) [coords\\_](#)
- [std::vector](#)< [std::vector](#)< [RoomTile](#) \* > > [tileVector\\_](#)
- [sf::RenderTexture](#) [roomTexture](#)
- [sf::Sprite](#) [roomBackground](#)
- [std::vector](#)< [MonsterSP](#) > [monsters\\_](#)
- [MonsterSpawner](#) \* [spawner\\_](#)
- [std::vector](#)< [Potion](#) \* > [potions\\_](#)
- bool [cleared\\_](#)
- bool [visited\\_](#)
- [vector](#)< [Direction](#) > [directionsLeft\\_](#)

### 7.31.1 Detailed Description

Class representing a room of a dungeon, usually includes a monsterspawner.

### 7.31.2 Constructor & Destructor Documentation

#### 7.31.2.1 RoomInstance() [1/3]

```
RoomInstance::RoomInstance (
    sf::Vector2u window_size,
    sf::Vector2i coords )
```

Construct a new Room Instance object.

##### Parameters

<i>window_size</i>	TO BE CHANGED takes the window size to create a correctly sized room
<i>coords</i>	the map coordinate of the room, follows the cartesian coordinate system

#### 7.31.2.2 RoomInstance() [2/3]

```
RoomInstance::RoomInstance (
    sf::Vector2u window_size,
    sf::Vector2i coords,
    MonsterSpawner * spawner )
```

#### 7.31.2.3 RoomInstance() [3/3]

```
RoomInstance::RoomInstance ( ) [default]
```

#### 7.31.2.4 ~RoomInstance()

```
RoomInstance::~~RoomInstance ( ) [virtual]
```

### 7.31.3 Member Function Documentation



### 7.31.3.1 AddPotion()

```
void RoomInstance::AddPotion (
    Potion * potion )
```

### 7.31.3.2 Connect()

```
void RoomInstance::Connect (
    Direction dir,
    RoomInstance * room )
```

Connects two rooms to each other.

#### Parameters

<i>dir</i>	Direction where the new room resides
<i>room</i>	The new room

#### Note

Also adds the rooms as a connection on the argument room's side

### 7.31.3.3 CreateExit()

```
void RoomInstance::CreateExit (
    Direction direction )
```

Takes down a 2-wide in wall (replaces walls with floortiles) to be able to walk between two rooms.

#### Parameters

<i>direction</i>	The direction of the exit
------------------	---------------------------

### 7.31.3.4 deleteMonster()

```
void RoomInstance::deleteMonster (
    MonsterSP m )
```

### 7.31.3.5 Enter()

```
void RoomInstance::Enter (
    PlayerPS player,
    Direction direction ) [virtual]
```

Function called when entering a room, needs player as parameter to perform some calculations about room difficulty.

#### Parameters

<i>player</i>	
<i>direction</i>	the direction we enter from

Reimplemented in [BossRoom](#).

### 7.31.3.6 Exit()

```
void RoomInstance::Exit ( )
```

Function called when exiting a room modifies cleared\_ and visited\_ booleans.

### 7.31.3.7 GetCoords()

```
sf::Vector2i RoomInstance::GetCoords ( ) [inline]
```

Get the coordinates of this room on the map.

#### Returns

sf::Vector2i

### 7.31.3.8 GetEntranceInDirection()

```
sf::Vector2u RoomInstance::GetEntranceInDirection (
    Direction direction )
```

### 7.31.3.9 GetMonsters()

```
std::vector< MonsterSP > & RoomInstance::GetMonsters ( )
```

**7.31.3.10 GetPotions()**

```
std::vector< Potion * > & RoomInstance::GetPotions ( )
```

**7.31.3.11 GetRoomInDir()**

```
RoomInstance * RoomInstance::GetRoomInDir (
    Direction dir )
```

Returns the room found in the wanted direction, nullptr if it is not found.

**Parameters**

<i>dir</i>	The direction
------------	---------------

**Returns**

RoomInstance\*

**7.31.3.12 getRoomTilesAt()**

```
std::vector< RoomTile * > RoomInstance::getRoomTilesAt (
    sf::FloatRect bounds )
```

**7.31.3.13 getTiles()**

```
std::vector< std::vector< RoomTile * > > RoomInstance::getTiles ( ) const
```

**7.31.3.14 HasDirectionsLeft()**

```
bool RoomInstance::HasDirectionsLeft ( )
```

Function used for Generating the dungeon map. Checks if this roomInstance has any direction that does not already contain an other roomInstance.

**Returns**

true Meaning that this roomInstance can be used to generate a new roomInstance

false Meaning that this roomInstance cannot be used to generate a new roomInstance

### 7.31.3.15 IsCleared()

```
bool RoomInstance::IsCleared ( )
```

Function the check if the roomInstance is cleared of Monsters.

#### Returns

true Meaning that this roomInstance is cleared of Monsters  
false Meaning that this roomInstance is not cleared of Monsters

### 7.31.3.16 IsVisisted()

```
bool RoomInstance::IsVisisted ( ) [inline]
```

### 7.31.3.17 monsterCleared()

```
bool RoomInstance::monsterCleared ( )
```

### 7.31.3.18 positionIsPenetratable()

```
bool RoomInstance::positionIsPenetratable (
    sf::FloatRect bounds )
```

### 7.31.3.19 positionIsWalkable()

```
bool RoomInstance::positionIsWalkable (
    sf::FloatRect bounds )
```

### 7.31.3.20 RemoveDirection()

```
void RoomInstance::RemoveDirection (
    Direction dir )
```

Function used for Generating the dungeon map. Removes the Direction dir from the directionsLeft\_ vector.

## Parameters

<i>dir</i>	The Direction to remove.
------------	--------------------------

**7.31.3.21 RemoveRandomDirection()**

```
Direction RoomInstance::RemoveRandomDirection ( )
```

Function used for Generating the dungeon map. Removes and returns a random Direction from the directionsLeft↵\_ vector.

## Returns

Direction The Direction that was randomly chosen.

**7.31.3.22 Render()**

```
void RoomInstance::Render (
    sf::RenderTarget * target )
```

Renders the room.

## Parameters

<i>target</i>	Where to render the room. About always game window
---------------	--

**7.31.3.23 renderSpriteBackground()**

```
void RoomInstance::renderSpriteBackground ( )
```

Renders the rooms background to be a static backdrop, a very expensive operation.

**7.31.3.24 setTiles()**

```
void RoomInstance::setTiles ( ) [protected], [virtual]
```

Helper, Sets all the roomtiles of the roominstance.

## 7.31.4 Member Data Documentation

### 7.31.4.1 cleared\_

```
bool RoomInstance::cleared_ [protected]
```

### 7.31.4.2 coords\_

```
sf::Vector2i RoomInstance::coords_ [protected]
```

### 7.31.4.3 directionsLeft\_

```
vector<Direction> RoomInstance::directionsLeft_ [protected]
```

### 7.31.4.4 monsters\_

```
std::vector<MonsterSP> RoomInstance::monsters_ [protected]
```

### 7.31.4.5 potions\_

```
std::vector<Potion*> RoomInstance::potions_ [protected]
```

### 7.31.4.6 roomBackground

```
sf::Sprite RoomInstance::roomBackground [protected]
```

### 7.31.4.7 roomSize\_

```
sf::Vector2u RoomInstance::roomSize_ [protected]
```

### 7.31.4.8 roomTexture

```
sf::RenderTexture RoomInstance::roomTexture [protected]
```

### 7.31.4.9 spawner\_

```
MonsterSpawner* RoomInstance::spawner_ [protected]
```

### 7.31.4.10 tileVector\_

```
std::vector<std::vector<RoomTile*> > RoomInstance::tileVector_ [protected]
```

### 7.31.4.11 visited\_

```
bool RoomInstance::visited_ [protected]
```

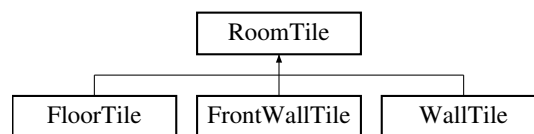
The documentation for this class was generated from the following files:

- [src/Dungeon/roomInstance.hpp](#)
- [src/Dungeon/roomInstance.cpp](#)

## 7.32 RoomTile Class Reference

```
#include <roomTile.hpp>
```

Inheritance diagram for RoomTile:



### Public Member Functions

- [RoomTile](#) (std::string texture, float x, float y, bool walkable, bool penetratable)  
Construct a new Room Tile object.
- const sf::Vector2f [getSize](#) () const
- const sf::Vector2f & [getPosition](#) () const
- const sf::FloatRect [getBoundingBox](#) () const
- const sf::Sprite & [getSprite](#) () const
- bool [isWalkable](#) () const
- bool [isPenetratable](#) () const

## 7.32.1 Constructor & Destructor Documentation

### 7.32.1.1 RoomTile()

```
RoomTile::RoomTile (
    std::string texture,
    float x,
    float y,
    bool walkable,
    bool penetratable )
```

Construct a new Room Tile object.

#### Parameters

<i>texture</i>	texture of the tile
<i>x</i>	location in x-axis
<i>y</i>	location in y-axis
<i>walkable</i>	is it a walkable tile or not
<i>penetratable</i>	can projectiles penetrate the tile

## 7.32.2 Member Function Documentation

### 7.32.2.1 getBoundingBox()

```
const sf::FloatRect RoomTile::getBoundingBox ( ) const
```

### 7.32.2.2 getPosition()

```
const sf::Vector2f & RoomTile::getPosition ( ) const
```

### 7.32.2.3 getSize()

```
const sf::Vector2f RoomTile::getSize ( ) const
```



#### 7.32.2.4 getSprite()

```
const sf::Sprite & RoomTile::getSprite ( ) const
```

#### 7.32.2.5 isPenetratable()

```
bool RoomTile::isPenetratable ( ) const
```

#### 7.32.2.6 isWalkable()

```
bool RoomTile::isWalkable ( ) const
```

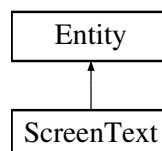
The documentation for this class was generated from the following files:

- [src/Dungeon/Tiles/roomTile.hpp](#)
- [src/Dungeon/Tiles/roomTile.cpp](#)

## 7.33 ScreenText Class Reference

```
#include <ScreenText.hpp>
```

Inheritance diagram for ScreenText:



### Public Member Functions

- [ScreenText](#) (const std::string &textLocation, sf::Vector2f textPos, sf::Vector2f textDims)
- [~ScreenText](#) ()

### Additional Inherited Members

#### 7.33.1 Constructor & Destructor Documentation

### 7.33.1.1 ScreenText()

```
ScreenText::ScreenText (
    const std::string & textLocation,
    sf::Vector2f textPos,
    sf::Vector2f textDims )
```

### 7.33.1.2 ~ScreenText()

```
ScreenText::~ScreenText ( ) [inline]
```

The documentation for this class was generated from the following files:

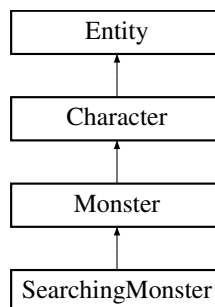
- [src/Utility/ScreenText.hpp](#)
- [src/Utility/ScreenText.cpp](#)

## 7.34 SearchingMonster Class Reference

[SearchingMonster](#) is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player.

```
#include <SearchingMonster.hpp>
```

Inheritance diagram for SearchingMonster:



### Public Member Functions

- [SearchingMonster](#) ([PlayerPS](#) player, float xPos, float yPos)
- [SearchingMonster](#) ([PlayerPS](#) player, sf::Vector2f pos)
- [~SearchingMonster](#) ()
- virtual std::list< [ProjectileUP](#) > [Attack](#) ()
- virtual bool [Move](#) (float dt)
- void [initVariables](#) ()

## Additional Inherited Members

### 7.34.1 Detailed Description

[SearchingMonster](#) is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player.

### 7.34.2 Constructor & Destructor Documentation

#### 7.34.2.1 SearchingMonster() [1/2]

```
SearchingMonster::SearchingMonster (
    PlayerPS player,
    float xPos,
    float yPos )
```

#### 7.34.2.2 SearchingMonster() [2/2]

```
SearchingMonster::SearchingMonster (
    PlayerPS player,
    sf::Vector2f pos )
```

#### 7.34.2.3 ~SearchingMonster()

```
SearchingMonster::~~SearchingMonster ( )
```

### 7.34.3 Member Function Documentation

#### 7.34.3.1 Attack()

```
std::list< ProjectileUP > SearchingMonster::Attack ( ) [virtual]
```

Implements [Monster](#).

### 7.34.3.2 initVariables()

```
void SearchingMonster::initVariables ( )
```

### 7.34.3.3 Move()

```
bool SearchingMonster::Move (
    float dt ) [virtual]
```

Implements [Monster](#).

The documentation for this class was generated from the following files:

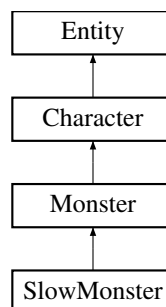
- src/Actors/Monsters/[SearchingMonster.hpp](#)
- src/Actors/Monsters/[SearchingMonster.cpp](#)

## 7.35 SlowMonster Class Reference

[SlowMonster](#) is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposely inaccurate.

```
#include <SlowMonster.hpp>
```

Inheritance diagram for SlowMonster:



### Public Member Functions

- [SlowMonster](#) ([PlayerPS](#) player, float xPos, float yPos)
- [SlowMonster](#) ([PlayerPS](#) player, sf::Vector2f pos)
- [~SlowMonster](#) ()
- virtual std::list< [ProjectileUP](#) > [Attack](#) ()
- virtual bool [Move](#) (float dt)
- void [initVariables](#) ()

## Additional Inherited Members

### 7.35.1 Detailed Description

[SlowMonster](#) is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposely inaccurate.

### 7.35.2 Constructor & Destructor Documentation

#### 7.35.2.1 SlowMonster() [1/2]

```
SlowMonster::SlowMonster (
    PlayerPS player,
    float xPos,
    float yPos )
```

#### 7.35.2.2 SlowMonster() [2/2]

```
SlowMonster::SlowMonster (
    PlayerPS player,
    sf::Vector2f pos )
```

#### 7.35.2.3 ~SlowMonster()

```
SlowMonster::~~SlowMonster ( )
```

### 7.35.3 Member Function Documentation

#### 7.35.3.1 Attack()

```
std::list< ProjectileUP > SlowMonster::Attack ( ) [virtual]
```

Implements [Monster](#).

### 7.35.3.2 initVariables()

```
void SlowMonster::initVariables ( )
```

### 7.35.3.3 Move()

```
bool SlowMonster::Move (
    float dt ) [virtual]
```

Implements [Monster](#).

The documentation for this class was generated from the following files:

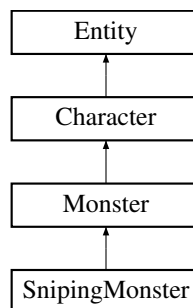
- src/Actors/Monsters/[SlowMonster.hpp](#)
- src/Actors/Monsters/[SlowMonster.cpp](#)

## 7.36 SnipingMonster Class Reference

[SnipingMonster](#) never moves but shoots projectile towards the player with a really big range.

```
#include <SnipingMonster.hpp>
```

Inheritance diagram for SnipingMonster:



### Public Member Functions

- [SnipingMonster](#) ([PlayerPS](#) player, float xPos, float yPos)
- [SnipingMonster](#) ([PlayerPS](#) player, sf::Vector2f pos)
- [~SnipingMonster](#) ()
- virtual std::list< [ProjectileUP](#) > [Attack](#) ()
- virtual bool [Move](#) (float dt)
- void [initVariables](#) ()

### Additional Inherited Members

#### 7.36.1 Detailed Description

[SnipingMonster](#) never moves but shoots projectile towards the player with a really big range.

## 7.36.2 Constructor & Destructor Documentation

### 7.36.2.1 SnipingMonster() [1/2]

```
SnipingMonster::SnipingMonster (
    PlayerPS player,
    float xPos,
    float yPos )
```

### 7.36.2.2 SnipingMonster() [2/2]

```
SnipingMonster::SnipingMonster (
    PlayerPS player,
    sf::Vector2f pos )
```

### 7.36.2.3 ~SnipingMonster()

```
SnipingMonster::~SnipingMonster ( )
```

## 7.36.3 Member Function Documentation

### 7.36.3.1 Attack()

```
std::list< ProjectileUP > SnipingMonster::Attack ( ) [virtual]
```

Implements [Monster](#).

### 7.36.3.2 initVariables()

```
void SnipingMonster::initVariables ( )
```

### 7.36.3.3 Move()

```
bool SnipingMonster::Move (
    float dt ) [virtual]
```

Implements [Monster](#).

The documentation for this class was generated from the following files:

- src/Actors/Monsters/[SnipingMonster.hpp](#)
- src/Actors/Monsters/[SnipingMonster.cpp](#)

## 7.37 SoundEffect Class Reference

```
#include <SoundEffects.hpp>
```

### Public Member Functions

- [SoundEffect](#) (const std::string &SoundEffectFilename)
- [~SoundEffect](#) ()
- void [PlaySound](#) ()

### 7.37.1 Constructor & Destructor Documentation

#### 7.37.1.1 SoundEffect()

```
SoundEffect::SoundEffect (
    const std::string & SoundEffectFilename )
```

#### 7.37.1.2 ~SoundEffect()

```
SoundEffect::~~SoundEffect ( ) [inline]
```

### 7.37.2 Member Function Documentation



### 7.37.2.1 PlaySound()

```
void SoundEffect::PlaySound ( )
```

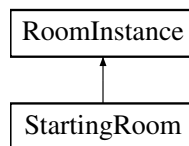
The documentation for this class was generated from the following files:

- src/Utility/Sounds/[SoundEffects.hpp](#)
- src/Utility/Sounds/[SoundEffects.cpp](#)

## 7.38 StartingRoom Class Reference

```
#include <startingRoom.hpp>
```

Inheritance diagram for StartingRoom:



### Public Member Functions

- [StartingRoom](#) (sf::Vector2u window\_size, sf::Vector2i choords)  
*Construct a new Starting Room object from window size and set its choordinates.*
- [StartingRoom](#) ()
- [~StartingRoom](#) ()
- virtual void [setTiles](#) (sf::Vector2u window\_size)

### Additional Inherited Members

## 7.38.1 Constructor & Destructor Documentation

### 7.38.1.1 StartingRoom() [1/2]

```
StartingRoom::StartingRoom (
    sf::Vector2u window_size,
    sf::Vector2i choords )
```

Construct a new Starting Room object from window size and set its choordinates.

#### Parameters

<i>window_size</i>	size of window
<i>choords</i>	choordinates of the starting room in map

### 7.38.1.2 StartingRoom() [2/2]

```
StartingRoom::StartingRoom ( ) [inline]
```

### 7.38.1.3 ~StartingRoom()

```
StartingRoom::~~StartingRoom ( ) [inline]
```

## 7.38.2 Member Function Documentation

### 7.38.2.1 setTiles()

```
void StartingRoom::setTiles (
    sf::Vector2u window_size ) [virtual]
```

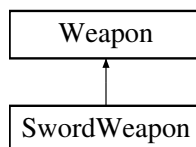
The documentation for this class was generated from the following files:

- src/Dungeon/specialrooms/[startingRoom.hpp](#)
- src/Dungeon/specialrooms/[startingRoom.cpp](#)

## 7.39 SwordWeapon Class Reference

```
#include <SwordWeapon.hpp>
```

Inheritance diagram for SwordWeapon:



### Public Member Functions

- [SwordWeapon](#) (int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const std::string &spriteLocation)
- virtual [~SwordWeapon](#) ()
- virtual [ProjectileUP Use](#) (Vector2f dir, Vector2f origin)

## Additional Inherited Members

### 7.39.1 Constructor & Destructor Documentation

#### 7.39.1.1 SwordWeapon()

```
SwordWeapon::SwordWeapon (
    int damage,
    int range,
    int rateOfFire,
    float projectileSpeed,
    Vector2f projectileSize,
    const std::string & spriteLocation )
```

#### 7.39.1.2 ~SwordWeapon()

```
virtual SwordWeapon::~~SwordWeapon ( ) [inline], [virtual]
```

### 7.39.2 Member Function Documentation

#### 7.39.2.1 Use()

```
ProjectileUP SwordWeapon::Use (
    Vector2f dir,
    Vector2f origin ) [virtual]
```

Implements [Weapon](#).

The documentation for this class was generated from the following files:

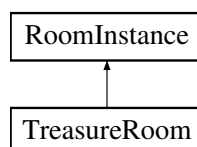
- [src/Combat/Weapons/SwordWeapon.hpp](#)
- [src/Combat/Weapons/SwordWeapon.cpp](#)

## 7.40 TreasureRoom Class Reference

Unused, was going to represent a room that includes a treasure and no monsters.

```
#include <TreasureRoom.hpp>
```

Inheritance diagram for TreasureRoom:



## Public Member Functions

- [TreasureRoom](#) (sf::Vector2u window\_size, sf::Vector2i choords)
- [TreasureRoom](#) ()
- [~TreasureRoom](#) ()
- virtual void [setTiles](#) (sf::Vector2u window\_size)

## Additional Inherited Members

### 7.40.1 Detailed Description

Unused, was going to represent a room that includes a treasure and no monsters.

### 7.40.2 Constructor & Destructor Documentation

#### 7.40.2.1 [TreasureRoom\(\)](#) [1/2]

```
TreasureRoom::TreasureRoom (
    sf::Vector2u window_size,
    sf::Vector2i choords )
```

#### 7.40.2.2 [TreasureRoom\(\)](#) [2/2]

```
TreasureRoom::TreasureRoom ( ) [inline]
```

#### 7.40.2.3 [~TreasureRoom\(\)](#)

```
TreasureRoom::~~TreasureRoom ( ) [inline]
```

### 7.40.3 Member Function Documentation

#### 7.40.3.1 [setTiles\(\)](#)

```
void TreasureRoom::setTiles (
    sf::Vector2u window_size ) [virtual]
```

The documentation for this class was generated from the following files:

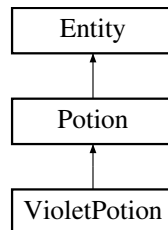
- src/Dungeon/specialrooms/[TreasureRoom.hpp](#)
- src/Dungeon/specialrooms/[TreasureRoom.cpp](#)

## 7.41 VioletPotion Class Reference

A violet potion. Takes a position as parameter and the colour and healing effect is set automatically.

```
#include <HealthPotions.hpp>
```

Inheritance diagram for VioletPotion:



### Public Member Functions

- [VioletPotion](#) (sf::Vector2f pos)

### Additional Inherited Members

#### 7.41.1 Detailed Description

A violet potion. Takes a position as parameter and the colour and healing effect is set automatically.

#### 7.41.2 Constructor & Destructor Documentation

##### 7.41.2.1 VioletPotion()

```
VioletPotion::VioletPotion (
    sf::Vector2f pos ) [inline]
```

The documentation for this class was generated from the following file:

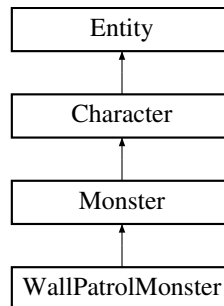
- src/Combat/Health/[HealthPotions.hpp](#)

## 7.42 WallPatrolMonster Class Reference

[WallPatrolMonster](#) walks along the walls of the room patrolling. Then when the player is in its range it shoots projectiles towards it.

```
#include <WallPatrolMonster.hpp>
```

Inheritance diagram for WallPatrolMonster:



### Public Member Functions

- [WallPatrolMonster](#) ([PlayerPS](#) player, float xPos, float yPos)
- [WallPatrolMonster](#) ([PlayerPS](#) player, sf::Vector2f pos)
- [~WallPatrolMonster](#) ()
- virtual std::list< [ProjectileUP](#) > [Attack](#) ()
- virtual bool [Move](#) (float dt)
- void [initVariables](#) ()

### Additional Inherited Members

#### 7.42.1 Detailed Description

[WallPatrolMonster](#) walks along the walls of the room patrolling. Then when the player is in its range it shoots projectiles towards it.

#### 7.42.2 Constructor & Destructor Documentation

##### 7.42.2.1 WallPatrolMonster() [1/2]

```
WallPatrolMonster::WallPatrolMonster (
    PlayerPS player,
    float xPos,
    float yPos )
```

### 7.42.2.2 WallPatrolMonster() [2/2]

```
WallPatrolMonster::WallPatrolMonster (
    PlayerPS player,
    sf::Vector2f pos )
```

### 7.42.2.3 ~WallPatrolMonster()

```
WallPatrolMonster::~~WallPatrolMonster ( )
```

## 7.42.3 Member Function Documentation

### 7.42.3.1 Attack()

```
std::list< ProjectileUP > WallPatrolMonster::Attack ( ) [virtual]
```

Implements [Monster](#).

### 7.42.3.2 initVariables()

```
void WallPatrolMonster::initVariables ( )
```

### 7.42.3.3 Move()

```
bool WallPatrolMonster::Move (
    float dt ) [virtual]
```

Implements [Monster](#).

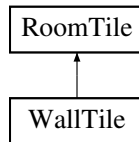
The documentation for this class was generated from the following files:

- [src/Actors/Monsters/WallPatrolMonster.hpp](#)
- [src/Actors/Monsters/WallPatrolMonster.cpp](#)

## 7.43 WallTile Class Reference

```
#include <roomTile.hpp>
```

Inheritance diagram for WallTile:



### Public Member Functions

- [WallTile](#) (std::string texture, float x, float y)

### 7.43.1 Constructor & Destructor Documentation

#### 7.43.1.1 WallTile()

```
WallTile::WallTile (  
    std::string texture,  
    float x,  
    float y ) [inline]
```

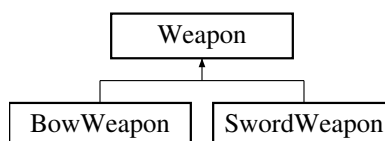
The documentation for this class was generated from the following file:

- src/Dungeon/Tiles/[roomTile.hpp](#)

## 7.44 Weapon Class Reference

```
#include <Weapon.hpp>
```

Inheritance diagram for Weapon:





## Public Member Functions

- [Weapon](#) (int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const std::string &spriteLocation)
- virtual [~Weapon](#) ()
- virtual [ProjectileUP Use](#) (Vector2f dir, Vector2f origin)=0
- void [AddPowerUp](#) ([PowerUp](#) \*up)
- float [GetAttackCooldown](#) ()
- float [GetRange](#) ()
- void [SetTextureRect](#) (sf::IntRect rect)
- void [BoostDamageValue](#) ()
- void [UnBoostDamageValue](#) ()

## Protected Member Functions

- int [getPowerUpCount](#) ()

## Protected Attributes

- int [defaultDamage\\_](#)
- int [currentDamage\\_](#)
- float [damageBoostModifier](#) = 1.1f
- int [range\\_](#)
- float [projectileSpeed\\_](#)
- Vector2f [projectileSize\\_](#)
- int [speed\\_](#)
- bool [penetrates\\_](#) = false
- const int [maxPowerUps](#) = 3
- Sprite [sprite\\_](#)
- Texture [texture\\_](#)
- vector< [PowerUp](#) \* > [powerUps\\_](#)
- float [cooldown\\_](#)
- float [attackCooldownLength\\_](#)
- float [attackCooldownLeft](#)

### 7.44.1 Constructor & Destructor Documentation

#### 7.44.1.1 [Weapon\(\)](#)

```
Weapon::Weapon (
    int damage,
    int range,
    int rateOfFire,
    float projectileSpeed,
    Vector2f projectileSize,
    const std::string & spriteLocation )
```

#### 7.44.1.2 ~Weapon()

```
virtual Weapon::~~Weapon ( ) [inline], [virtual]
```

### 7.44.2 Member Function Documentation

#### 7.44.2.1 AddPowerUp()

```
void Weapon::AddPowerUp (
    PowerUp * up )
```

#### 7.44.2.2 BoostDamageValue()

```
void Weapon::BoostDamageValue ( )
```

#### 7.44.2.3 GetAttackCooldown()

```
float Weapon::GetAttackCooldown ( ) [inline]
```

#### 7.44.2.4 getPowerUpCount()

```
int Weapon::getPowerUpCount ( ) [protected]
```

#### 7.44.2.5 GetRange()

```
float Weapon::GetRange ( ) [inline]
```

#### 7.44.2.6 SetTextureRect()

```
void Weapon::SetTextureRect (
    sf::IntRect rect )
```

### 7.44.2.7 UnBoostDamageValue()

```
void Weapon::UnBoostDamageValue ( )
```

### 7.44.2.8 Use()

```
virtual ProjectileUP Weapon::Use (
    Vector2f dir,
    Vector2f origin ) [pure virtual]
```

Implemented in [BowWeapon](#), and [SwordWeapon](#).

## 7.44.3 Member Data Documentation

### 7.44.3.1 attackCooldownLeft

```
float Weapon::attackCooldownLeft [protected]
```

### 7.44.3.2 attackCooldownLength\_

```
float Weapon::attackCooldownLength_ [protected]
```

### 7.44.3.3 cooldown\_

```
float Weapon::cooldown_ [protected]
```

### 7.44.3.4 currentDamage\_

```
int Weapon::currentDamage_ [protected]
```

### 7.44.3.5 damageBoostModifier

```
float Weapon::damageBoostModifier = 1.1f [protected]
```

#### 7.44.3.6 defaultDamage\_

```
int Weapon::defaultDamage_ [protected]
```

#### 7.44.3.7 maxPowerUps

```
const int Weapon::maxPowerUps = 3 [protected]
```

#### 7.44.3.8 penetrates\_

```
bool Weapon::penetrates_ = false [protected]
```

#### 7.44.3.9 powerUps\_

```
vector<PowerUp*> Weapon::powerUps_ [protected]
```

#### 7.44.3.10 projectileSize\_

```
Vector2f Weapon::projectileSize_ [protected]
```

#### 7.44.3.11 projectileSpeed\_

```
float Weapon::projectileSpeed_ [protected]
```

#### 7.44.3.12 range\_

```
int Weapon::range_ [protected]
```

#### 7.44.3.13 speed\_

```
int Weapon::speed_ [protected]
```

## 7.44.3.14 sprite\_

```
Sprite Weapon::sprite_ [protected]
```

## 7.44.3.15 texture\_

```
Texture Weapon::texture_ [protected]
```

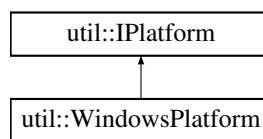
The documentation for this class was generated from the following files:

- src/Combat/Weapons/[Weapon.hpp](#)
- src/Combat/Weapons/[Weapon.cpp](#)

## 7.45 util::WindowsPlatform Struct Reference

```
#include <WindowsPlatform.hpp>
```

Inheritance diagram for util::WindowsPlatform:



## Public Member Functions

- [WindowsPlatform](#) ()
- void [setIcon](#) (const sf::WindowHandle &inHandle) final
- void [toggleFullscreen](#) (const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const bool inWindowed, const sf::Vector2u &inResolution) final
- float [getScreenScalingFactor](#) (const sf::WindowHandle &inHandle) final
- int [getRefreshRate](#) (const sf::WindowHandle &inHandle) final

## 7.45.1 Constructor &amp; Destructor Documentation

## 7.45.1.1 WindowsPlatform()

```
util::WindowsPlatform::WindowsPlatform ( )
```

## 7.45.2 Member Function Documentation

### 7.45.2.1 getRefreshRate()

```
int util::WindowsPlatform::getRefreshRate (
    const sf::WindowHandle & inHandle ) [final], [virtual]
```

Implements [util::IPlatform](#).

### 7.45.2.2 getScreenScalingFactor()

```
float util::WindowsPlatform::getScreenScalingFactor (
    const sf::WindowHandle & inHandle ) [final], [virtual]
```

Implements [util::IPlatform](#).

### 7.45.2.3 setIcon()

```
void util::WindowsPlatform::setIcon (
    const sf::WindowHandle & inHandle ) [final], [virtual]
```

Implements [util::IPlatform](#).

### 7.45.2.4 toggleFullscreen()

```
void util::WindowsPlatform::toggleFullscreen (
    const sf::WindowHandle & inHandle,
    const sf::Uint32 inStyle,
    const bool inWindowed,
    const sf::Vector2u & inResolution ) [final], [virtual]
```

Implements [util::IPlatform](#).

The documentation for this struct was generated from the following file:

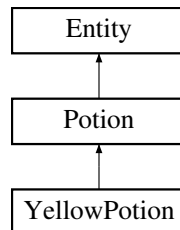
- [src/Platform/Win32/WindowsPlatform.hpp](#)

## 7.46 YellowPotion Class Reference

A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically.

```
#include <HealthPotions.hpp>
```

Inheritance diagram for YellowPotion:



### Public Member Functions

- [YellowPotion](#) (sf::Vector2f pos)

### Additional Inherited Members

#### 7.46.1 Detailed Description

A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically.

#### 7.46.2 Constructor & Destructor Documentation

##### 7.46.2.1 YellowPotion()

```
YellowPotion::YellowPotion (
    sf::Vector2f pos ) [inline]
```

The documentation for this class was generated from the following file:

- src/Combat/Health/[HealthPotions.hpp](#)





## Chapter 8

# File Documentation

### 8.1 src/Actors/character.cpp File Reference

```
#include "Actors/character.hpp"  
#include "Utility/SpriteHelper.hpp"
```

#### Macros

- `#define C_X 8`
- `#define C_PIXELS_X 44`
- `#define C_PIXELS_delta 20`
- `#define C_PIXELS_Y 64`
- `#define C_SCALE 2`

#### 8.1.1 Macro Definition Documentation

##### 8.1.1.1 C\_PIXELS\_delta

```
#define C_PIXELS_delta 20
```

##### 8.1.1.2 C\_PIXELS\_X

```
#define C_PIXELS_X 44
```

### 8.1.1.3 C\_PIXELS\_Y

```
#define C_PIXELS_Y 64
```

### 8.1.1.4 C\_SCALE

```
#define C_SCALE 2
```

### 8.1.1.5 C\_X

```
#define C_X 8
```

## 8.2 src/Actors/character.hpp File Reference

```
#include "Combat/Projectile.hpp"
#include "Animation/Animationhandler.hpp"
#include "Combat/Weapons/Weapon.hpp"
#include "Utility/LevelUpSystem.hpp"
#include "Utility/RandomHelper.hpp"
#include "entity.hpp"
```

### Classes

- class [Character](#)  
*[Character](#) class that our player and monster inherits.*

### Macros

- #define [\\_CHARACTER\\_CLASS\\_](#)

### 8.2.1 Macro Definition Documentation

#### 8.2.1.1 \_CHARACTER\_CLASS\_

```
#define _CHARACTER_CLASS_
```

## 8.3 character.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #ifndef _CHARACTER_CLASS_
4 #define _CHARACTER_CLASS_
5
6 #include "Combat/Projectile.hpp"
7 // #include "Interfaces/ICollidable.hpp"
8 #include "Animation/Animationhandler.hpp"
9 #include "Combat/Weapons/Weapon.hpp"
10 #include "Utility/LevelUpSystem.hpp"
11 #include "Utility/RandomHelper.hpp"
12 #include "entity.hpp"
13
14 class Character : public Entity /*, public ICollidable*/ {
15 public:
16     Character(const std::string& filename, sf::Vector2f pos, bool animated = false);
17
18     virtual ~Character();
19
20     virtual void Update(float dt) = 0;
21     void Equip(Weapon* weapon);
22
23     void initVariables();
24     void TakeDamage(int value);
25     void Heal(int value);
26     int GetHitPoints() const;
27     bool IsAlive();
28     bool HasWeapon();
29     bool Idle();
30     bool Dead();
31     bool MoveLeft(float dt);
32     bool MoveRight(float dt);
33     bool MoveDown(float dt);
34     bool MoveUp(float dt);
35     // For subclasses, should be = 0
36     virtual bool Move(float)
37     {
38         return false;
39     }
40     void RevertMove();
41
42     void ResetAttackCooldown();
43     float GetAttackCooldownLeft() const { return attackCooldownLeft; };
44     float GetAttackCooldownLength() const { return attackCooldownLength; };
45     bool CanAttack;
46     int GetMaxHP();
47
48     void SetNormalSpeed(float value);
49     void ResetCharacterToBeAlive();
50     /*
51     // for ICollidable
52     virtual sf::FloatRect GetBoundingBox() { return sprite_.getGlobalBounds(); }
53     virtual void ProcessCollision(ICollidable* object);
54     virtual ICollidable::EntityType GetEntityType();
55     */
56
57 protected:
58     /*void GetHitBy(Projectile& projectile);*/
59
60     Weapon* weapon_;
61     Projectile::Type characterProjectileType_;
62
63     sf::Vector2f startPos;
64
65     int hitpoints_;
66     int currentMaxHitpoints_;
67     int defaultMaxHitpoints_;
68
69     bool hasAnimation_;
70     AnimationHandler animationHandler_;
71     float currentSpeed_;
72     float defaultSpeed_;
73     bool left_or_right_ = true;
74     void generalUpdate(float dt);
75     bool invincibility_frame_ = false;
76     float attackCooldownLength_;
77     float attackCooldownLeft;
78     void updateAttackCooldown(float dt);
79     std::list<ProjectileUP> emptyList();
80     std::list<ProjectileUP> shotProjectileList(sf::Vector2f aimPos);
81 };
82 #endif

```

## 8.4 src/Actors/Monsters/BossMonster.cpp File Reference

```
#include "BossMonster.hpp"
#include "math.h"
```

## 8.5 src/Actors/Monsters/BossMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "monster.hpp"
```

### Classes

- class [BossMonster](#)

*This is the boss monster. It is the most powerful monster and once the player kills it the game is won.*

### Macros

- `#define` [\\_BOSS\\_MONSTER\\_CLASS\\_](#)

## 8.5.1 Macro Definition Documentation

### 8.5.1.1 \_BOSS\_MONSTER\_CLASS\_

```
#define _BOSS_MONSTER_CLASS_
```

## 8.6 BossMonster.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _BOSS_MONSTER_CLASS_
4 #define _BOSS_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/SwordWeapon.hpp"
9 #include "monster.hpp"
10
11 class BossMonster : public Monster {
12 public:
13     BossMonster(PlayerPS player, float xPos, float yPos);
14     BossMonster(PlayerPS player, sf::Vector2f pos);
15     ~BossMonster();
16
17     virtual std::list<ProjectileUP> Attack();
```

```
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27     int currentDir_;
28
29     int attackStyle_ = 0;
30     int nofattackStyles_ = 3;
31
32     float angle_ = 0;
33     int nofBulletsInCircle_ = 10;
34     int nofBulletsToShootTowardsPlayer_ = 5;
35     int nofBulletsShot_ = 0;
36
37     sf::Clock attackLoopClock_;
38     float spritalAttackCooldownLength_ = 0.1f;
39     float normalAttackCooldownLength_ = 0.5f;
40
41     float durationUntilTurn_ = 0.5f;
42     float elapsedTurnTime_ = 0.0f;
43
44     void iterateAngle();
45     void iterateAttackStyle();
46 };
47
48 #endif
```

## 8.7 src/Actors/Monsters/monster.cpp File Reference

```
#include "monster.hpp"
```

## 8.8 src/Actors/Monsters/monster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Health/HealthPotions.hpp"
```

### Classes

- class [Monster](#)  
*[Monster](#) class, which all our other monsters inherit.*

### Macros

- #define [\\_MONSTER\\_CLASS\\_](#)

#### 8.8.1 Macro Definition Documentation

##### 8.8.1.1 [\\_MONSTER\\_CLASS\\_](#)

```
#define _MONSTER_CLASS_
```

## 8.9 monster.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #ifndef _MONSTER_CLASS_
4 #define _MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Health/HealthPotions.hpp"
13 class Monster : public Character {
14 public:
15     virtual ~Monster();
16
17     Player& GetPlayer() const;
18     virtual std::list<ProjectileUP> Attack() = 0;
19     virtual void Update(float);
20     virtual bool Move(float dt) = 0;
21     virtual void Render(sf::RenderTarget* target);
22     void initVariables();
23     void SetTarget(PlayerPS target);
29     Potion* ReturnPotion();
30
31 protected:
32     Monster(PlayerPS player, sf::Vector2f pos, const std::string& spriteFile);
33     Monster(PlayerPS player, float xPos, float yPos, const std::string& spriteFile);
34     std::shared_ptr<Player> player_;
35     sf::RectangleShape healthbar_;
36     float staticDamage_ = 5.0f;
37     float getDistanceToPlayer();
44     bool inRangeOfPlayer();
45     bool movedLastTick_;
46
47     bool moveTowardsPlayer(float dt);
52     void clampPosToRoom();
53 };
54
55 #endif

```

## 8.10 src/Actors/Monsters/MonsterSpawner/BossSpawner.cpp File Reference

```
#include "BossSpawner.hpp"
```

## 8.11 src/Actors/Monsters/MonsterSpawner/BossSpawner.hpp File Reference

```

#include "../BossMonster.hpp"
#include "MonsterSpawner.hpp"

```

### Classes

- class [BossSpawner](#)

### Macros

- #define [\\_BOSS\\_SPAWNER\\_CLASS\\_](#)

## 8.11.1 Macro Definition Documentation

### 8.11.1.1 `_BOSS_SPAWNER_CLASS_`

```
#define _BOSS_SPAWNER_CLASS_
```

## 8.12 BossSpawner.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _BOSS_SPAWNER_CLASS_
3 #define _BOSS_SPAWNER_CLASS_
4 #include "../BossMonster.hpp"
5 #include "MonsterSpawner.hpp"
6
7 class BossSpawner : public MonsterSpawner {
8 public:
9     BossSpawner()
10         : MonsterSpawner(1)
11     {
12     }
13
14     ~BossSpawner() { }
15     virtual MonsterSP SpawnMonster(sf::Vector2u roomSize, PlayerPS target);
16 };
17
18 #endif
```

## 8.13 src/Actors/Monsters/MonsterSpawner/MonsterSpawner.cpp File Reference

```
#include "MonsterSpawner.hpp"
#include <time.h>
```

## 8.14 src/Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp File Reference

```
#include "../BossMonster.hpp"
#include "../RandomMonster.hpp"
#include "../SearchingMonster.hpp"
#include "../SlowMonster.hpp"
#include "../SnipingMonster.hpp"
#include "../WallPatrolMonster.hpp"
```

### Classes

- class [MonsterSpawner](#)

## Macros

- `#define \_MONSTERSPAWNER\_CLASS\_`

## Typedefs

- `typedef std::shared_ptr<Monster> MonsterSP`
- `typedef std::shared_ptr<MonsterSpawner> MonsterSpawnerUP`

### 8.14.1 Macro Definition Documentation

#### 8.14.1.1 [\\_MONSTERSPAWNER\\_CLASS\\_](#)

```
#define \_MONSTERSPAWNER\_CLASS\_
```

### 8.14.2 Typedef Documentation

#### 8.14.2.1 [MonsterSP](#)

```
typedef std::shared_ptr<Monster> MonsterSP
```

#### 8.14.2.2 [MonsterSpawnerUP](#)

```
typedef std::shared_ptr<MonsterSpawner> MonsterSpawnerUP
```



## 8.15 MonsterSpawner.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #ifndef _MONSTERSPAWNER_CLASS_
3 #define _MONSTERSPAWNER_CLASS_
4 #include "../BossMonster.hpp"
5 #include "../RandomMonster.hpp"
6 #include "../SearchingMonster.hpp"
7 #include "../SlowMonster.hpp"
8 #include "../SnipingMonster.hpp"
9 #include "../WallPatrolMonster.hpp"
10
11 typedef std::shared_ptr<Monster> MonsterSP;
12
13 class MonsterSpawner {
14
15 public:
16     MonsterSpawner(uint monsterAmount)
17         : monsterCount_(monsterAmount)
18     {
19     }
20     MonsterSpawner()
21         : monsterCount_(0) {};
22
23     virtual ~MonsterSpawner() { }
24
25     // std::list<Monster*> SpawnMonsters(sf::Vector2u roomSize) const;
26     virtual MonsterSP SpawnMonster(sf::Vector2u roomSize, PlayerPS target) const;
27     void SetMonsterAmount(uint amount);
28     uint GetMonsterAmount() const;
29
30 protected:
31     MonsterSP getRandomMonster(PlayerPS target) const;
32     uint monsterCount_;
33     uint monsterTypeCount_ = 5; // update this when adding monsters
34 };
35
36 typedef std::shared_ptr<MonsterSpawner> MonsterSpawnerUP;
37
38 #endif

```

## 8.16 src/Actors/Monsters/RandomMonster.cpp File Reference

```
#include "RandomMonster.hpp"
```

## 8.17 src/Actors/Monsters/RandomMonster.hpp File Reference

```

#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "monster.hpp"

```

### Classes

- class [RandomMonster](#)

*RandomMonster* is a monster that moves randomly around and shoots projectiles towards the player.

### Macros

- #define [\\_RANDOM\\_MONSTER\\_CLASS\\_](#)

## 8.17.1 Macro Definition Documentation

### 8.17.1.1 `_RANDOM_MONSTER_CLASS_`

```
#define _RANDOM_MONSTER_CLASS_
```

## 8.18 RandomMonster.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _RANDOM_MONSTER_CLASS_
4 #define _RANDOM_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/SwordWeapon.hpp"
9 #include "monster.hpp"
10
15 class RandomMonster : public Monster {
16 public:
17     RandomMonster(PlayerPS player, float xPos, float yPos);
18     RandomMonster(PlayerPS player, sf::Vector2f pos);
19     ~RandomMonster();
20
21     virtual std::list<ProjectileUP> Attack();
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27     int currentDir_;
28     float durationUntilTurn_ = 0.5f;
29     float elapsedTurnTime_ = 0.0f;
30 };
31
32 #endif
```

## 8.19 src/Actors/Monsters/SearchingMonster.cpp File Reference

```
#include "SearchingMonster.hpp"
```

## 8.20 src/Actors/Monsters/SearchingMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "monster.hpp"
```

## Classes

- class [SearchingMonster](#)

*SearchingMonster* is our only monster that does not shoot projectile but rather directly decreases the healthpoints of the player. It as its name says always walks towards the player.

## Macros

- `#define _SEARCHING_MONSTER_CLASS_`

### 8.20.1 Macro Definition Documentation

#### 8.20.1.1 \_SEARCHING\_MONSTER\_CLASS\_

```
#define _SEARCHING_MONSTER_CLASS_
```

## 8.21 SearchingMonster.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _SEARCHING_MONSTER_CLASS_
4 #define _SEARCHING_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "monster.hpp"
9
14 class SearchingMonster : public Monster {
15 public:
16     SearchingMonster(PlayerPS player, float xPos, float yPos);
17     SearchingMonster(PlayerPS player, sf::Vector2f pos);
18     ~SearchingMonster();
19
20     virtual std::list<ProjectileUP> Attack();
21     virtual bool Move(float dt);
22
23     void initVariables();
24
25 private:
26     sf::Clock cooldown_;
27     std::string name_ = "Sir Chi";
28 };
29
30 #endif
```

## 8.22 src/Actors/Monsters/SlowMonster.cpp File Reference

```
#include "SlowMonster.hpp"
```

## 8.23 src/Actors/Monsters/SlowMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/BowWeapon.hpp"
#include "monster.hpp"
```

## Classes

- class [SlowMonster](#)

*SlowMonster* is a monster that slowly walks towards the player and rapidly shoots a lot of projectiles. The aim of slowmonster is purposely inaccurate.

## Macros

- `#define _SLOW_MONSTER_CLASS_`

### 8.23.1 Macro Definition Documentation

#### 8.23.1.1 \_SLOW\_MONSTER\_CLASS\_

```
#define _SLOW_MONSTER_CLASS_
```

## 8.24 SlowMonster.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _SLOW_MONSTER_CLASS_
4 #define _SLOW_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/BowWeapon.hpp"
9 #include "monster.hpp"
10
11
12
13
14
15 class SlowMonster : public Monster {
16 public:
17     SlowMonster(PlayerPS player, float xPos, float yPos);
18     SlowMonster(PlayerPS player, sf::Vector2f pos);
19     ~SlowMonster();
20
21     virtual std::list<ProjectileUP> Attack();
22     virtual bool Move(float dt);
23
24     void initVariables();
25
26 private:
27 };
28
29 #endif
```

## 8.25 src/Actors/Monsters/SnipingMonster.cpp File Reference

```
#include "SnipingMonster.hpp"
```

## 8.26 src/Actors/Monsters/SnipingMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/BowWeapon.hpp"
#include "monster.hpp"
```

### Classes

- class [SnipingMonster](#)  
*SnipingMonster* never moves but shoots projectile towards the player with a really big range.

### Macros

- #define [\\_SNIPING\\_MONSTER\\_CLASS\\_](#)

#### 8.26.1 Macro Definition Documentation

##### 8.26.1.1 [\\_SNIPING\\_MONSTER\\_CLASS\\_](#)

```
#define _SNIPING_MONSTER_CLASS_
```

## 8.27 SnipingMonster.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _SNIPING_MONSTER_CLASS_
4 #define _SNIPING_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/BowWeapon.hpp"
9 #include "monster.hpp"
10
11 class SnipingMonster : public Monster {
12 public:
13     SnipingMonster(PlayerPS player, float xPos, float yPos);
14     SnipingMonster(PlayerPS player, sf::Vector2f pos);
15     ~SnipingMonster();
16
17     virtual std::list<ProjectileUP> Attack();
18     virtual bool Move(float dt);
19
20     void initVariables();
21
22 private:
23 };
24
25 #endif
```

## 8.28 src/Actors/Monsters/WallPatrolMonster.cpp File Reference

```
#include "WallPatrolMonster.hpp"
```

## 8.29 src/Actors/Monsters/WallPatrolMonster.hpp File Reference

```
#include "Actors/character.hpp"
#include "Actors/player.hpp"
#include "Combat/Weapons/SwordWeapon.hpp"
#include "monster.hpp"
```

### Classes

- class [WallPatrolMonster](#)  
*WallPatrolMonster* walks along the walls of the room patrolling. Then when the player is in its range it shoots projectiles towards it.

### Macros

- #define [\\_WALL\\_PATROL\\_MONSTER\\_CLASS\\_](#)

### 8.29.1 Macro Definition Documentation

#### 8.29.1.1 \_WALL\_PATROL\_MONSTER\_CLASS\_

```
#define _WALL_PATROL_MONSTER_CLASS_
```

## 8.30 WallPatrolMonster.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _WALL_PATROL_MONSTER_CLASS_
4 #define _WALL_PATROL_MONSTER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Weapons/SwordWeapon.hpp"
9 #include "monster.hpp"
10
11 class WallPatrolMonster : public Monster {
12 public:
13     WallPatrolMonster(PlayerPS player, float xPos, float yPos);
14     WallPatrolMonster(PlayerPS player, sf::Vector2f pos);
15     ~WallPatrolMonster();
16
17     virtual std::list<ProjectileUP> Attack();
18     virtual bool Move(float dt);
19
20     void initVariables();
21
22 private:
23     int currentDir_ = 1;
24     void changeDirection();
25 };
26
27 #endif
```

## 8.31 src/Actors/player.cpp File Reference

```
#include "Actors/player.hpp"
```

## 8.32 src/Actors/player.hpp File Reference

```
#include "Actors/character.hpp"  
#include "Combat/Health/Potion.hpp"
```

### Classes

- class [Player](#)  
*[Player](#) class, our controlled character.*

### Macros

- #define [\\_PLAYER\\_CLASS\\_](#)

### Typedefs

- typedef std::shared\_ptr< [Player](#) > [PlayerPS](#)

#### 8.32.1 Macro Definition Documentation

##### 8.32.1.1 [\\_PLAYER\\_CLASS\\_](#)

```
#define _PLAYER_CLASS_
```

#### 8.32.2 Typedef Documentation

##### 8.32.2.1 [PlayerPS](#)

```
typedef std::shared_ptr<Player> PlayerPS
```

## 8.33 player.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #ifndef _PLAYER_CLASS_
4 #define _PLAYER_CLASS_
5
6 #include "Actors/character.hpp"
7 #include "Combat/Health/Potion.hpp"
12 class Player : public Character {
13 public:
14     Player();
15     ~Player() {};
16
17     virtual void Update(float dt);
22     void Dash();
23
30     std::list<ProjectileUP> Attack(sf::Vector2f aimPos);
31
32     void initVariables();
33
34     void ResetDashCooldown();
35
36     float GetDashCooldownLeft() const { return dashCooldownLeft_; };
37     float GetDashCooldownLength() const { return dashCooldownLength_; };
38
39     bool CanDash;
40     bool IsDashing;
46     void AddPotion(Potion* potion);
52     void UsePotion(const std::string& colour);
58     std::vector<Potion*> GetInventory() const;
59     void ClearInventory();
60
61 private:
62     int attacksBoosted_;
63     int dashesBoosted_;
64
65     float dashSpeed_;
66     float dashLengthBoostModifier = 2.0f;
67     float dashDefaultDurationLength_;
68     float dashCurrentDurationLength_;
69     float dashDurationLeft_;
70     bool deadAnimationPlayed = false;
71     float dt_time = 0;
72     float dashCooldownLength_;
73     float dashCooldownLeft_;
74     void updateDashCooldown(float dt);
75
76     std::vector<Potion*> inventory_;
77 };
78 typedef std::shared_ptr<Player> PlayerPS;
79
80 #endif

```

## 8.34 src/Animation/animation.cpp File Reference

```

#include "animation.hpp"
#include "Utility/SpriteHelper.hpp"

```

## 8.35 src/Animation/animation.hpp File Reference

### Classes

- class [Animation](#)

*Animation* build the characters animation frames from spritesheet and displays them to achive an animated sprite.



## Macros

- `#define` [\\_ANIMATION\\_](#)

### 8.35.1 Macro Definition Documentation

#### 8.35.1.1 `_ANIMATION_`

```
#define _ANIMATION_
```

## 8.36 animation.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _ANIMATION_
4 #define _ANIMATION_
5
6 class Animation {
7 public:
8     Animation() = default;
9     Animation(int x, int y, int width, int height, int spacing, const std::string& textureName);
10    ~Animation() { }
11    void AnimationToSprite(sf::Sprite& sprite) const;
12    void Update(float dt);
13
14 private:
15     static constexpr int nFrames_ = 5;
16     static constexpr float holdTime_ = 0.15f;
17     sf::Texture texture_;
18     sf::IntRect frames_[nFrames_];
19     int iFrame_ = 0;
20     float time_ = 0.0f;
21     void NextFrame();
22 };
23
24 #endif
```

## 8.37 src/Animation/Animationhandler.cpp File Reference

```
#include "Animationhandler.hpp"
```

## 8.38 src/Animation/Animationhandler.hpp File Reference

```
#include "animation.hpp"
```

## Classes

- class [AnimationHandler](#)

## Macros

- `#define \_ANIMATIONHANDLER\_CLASS\_`

## Typedefs

- `typedef std::shared_ptr< Animation > AnimationPS`

## Enumerations

- `enum AnimationIndex {  
    AnimationUp , AnimationDown , AnimationLeft , AnimationRight ,  
    AnimationIdle , AnimationDeath , Count }`

### 8.38.1 Macro Definition Documentation

#### 8.38.1.1 [\\_ANIMATIONHANDLER\\_CLASS\\_](#)

```
#define \_ANIMATIONHANDLER\_CLASS\_
```

### 8.38.2 Typedef Documentation

#### 8.38.2.1 [AnimationPS](#)

```
typedef std::shared_ptr<Animation> AnimationPS
```

### 8.38.3 Enumeration Type Documentation

#### 8.38.3.1 [AnimationIndex](#)

```
enum AnimationIndex
```

##### Enumerator

<a href="#">AnimationUp</a>	
<a href="#">AnimationDown</a>	
<a href="#">AnimationLeft</a>	
<a href="#">AnimationRight</a>	
<a href="#">AnimationIdle</a>	
<a href="#">AnimationDeath</a>	
<a href="#">Count</a>	

## 8.39 Animationhandler.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #ifndef _ANIMATIONHANDLER_CLASS_
3 #define _ANIMATIONHANDLER_CLASS_
4 #include "animation.hpp"
5
6 typedef std::shared_ptr<Animation> AnimationPS;
7
8 enum AnimationIndex {
9     AnimationUp,
10    AnimationDown,
11    AnimationLeft,
12    AnimationRight,
13    AnimationIdle,
14    AnimationDeath,
15    Count
16 };
17
18 class AnimationHandler {
19 public:
20     AnimationHandler()
21         : animations_(0) {};
22     ~AnimationHandler();
23     AnimationHandler(uint xOffset, uint yOffset, uint width, uint height, uint xSpacing, const
24         std::string& textureLocation, const std::string& deathTexture);
25     void setAnimation(AnimationIndex index);
26     Animation* getAnimation() const;
27
28 private:
29     std::vector<AnimationPS> animations_;
30     AnimationIndex currentAnimationIndex_ = AnimationIndex::AnimationDown;
31 };
32 #endif

```

## 8.40 src/Combat/CircularProjectile.hpp File Reference

### 8.41 CircularProjectile.hpp

[Go to the documentation of this file.](#)

## 8.42 src/Combat/Health/HealthPotions.hpp File Reference

```

#include "Actors/player.hpp"
#include "Potion.hpp"

```

### Classes

- class [GreenPotion](#)  
A green potion. Takes a position as parameter and the colour and healing effect is set automatically.
- class [RedPotion](#)  
A red potion. Takes a position as parameter and the colour and healing effect is set automatically.
- class [YellowPotion](#)  
A yellow potion. Takes a position as parameter and the colour and healing effect is set automatically.
- class [VioletPotion](#)  
A violet potion. Takes a position as parameter and the colour and healing effect is set automatically.

## Macros

- `#define _HEALHPOTIONS_CLASS_`

### 8.42.1 Macro Definition Documentation

#### 8.42.1.1 `_HEALHPOTIONS_CLASS_`

```
#define _HEALHPOTIONS_CLASS_
```

## 8.43 HealthPotions.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _HEALHPOTIONS_CLASS_
4 #define _HEALHPOTIONS_CLASS_
5 #include "Actors/player.hpp"
6 #include "Potion.hpp"
7
12 class GreenPotion : public Potion {
13 public:
14     GreenPotion(sf::Vector2f pos)
15         : Potion("content/sprites/potions/green_potion.png", pos, 10, "green")
16     {
17     }
18 };
19
24 class RedPotion : public Potion {
25 public:
26     RedPotion(sf::Vector2f pos)
27         : Potion("content/sprites/potions/red_potion.png", pos, 5, "red")
28     {
29     }
30 };
31
36 class YellowPotion : public Potion {
37 public:
38     YellowPotion(sf::Vector2f pos)
39         : Potion("content/sprites/potions/yellow_potion.png", pos, 15, "yellow")
40     {
41     }
42 };
43
48 class VioletPotion : public Potion {
49 public:
50     VioletPotion(sf::Vector2f pos)
51         : Potion("content/sprites/potions/violet_potion.png", pos, 50, "violet")
52     {
53     }
54 };
55
56 #endif
```

## 8.44 src/Combat/Health/Potion.cpp File Reference

```
#include "Potion.hpp"
```

## 8.45 src/Combat/Health/Potion.hpp File Reference

```
#include "entity.hpp"
```

### Classes

- class [Potion](#)

### Macros

- #define [\\_POTION\\_CLASS\\_](#)

### 8.45.1 Macro Definition Documentation

#### 8.45.1.1 [\\_POTION\\_CLASS\\_](#)

```
#define _POTION_CLASS_
```

## 8.46 Potion.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _POTION_CLASS_
4 #define _POTION_CLASS_
5 #include "entity.hpp"
6
7 class Potion : public Entity {
8 public:
9     Potion(const std::string& spritefile, sf::Vector2f pos, int healthIncrease, const std::string&
        colour);
10
11     virtual ~Potion() {};
12
13     const std::string& GetColour() const;
14
15     int GetHealthIncrease() const;
16
17 protected:
18     int healthIncrease_;
19     std::string colour_;
20 };
21
22 #endif
```

## 8.47 src/Combat/PowerUps/PowerUp.hpp File Reference

### Classes

- class [PowerUp](#)

## Macros

- `#define` [\\_POWERUP\\_CLASS\\_](#)

### 8.47.1 Macro Definition Documentation

#### 8.47.1.1 [\\_POWERUP\\_CLASS\\_](#)

```
#define _POWERUP_CLASS_
```

## 8.48 PowerUp.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _POWERUP_CLASS_
3 #define _POWERUP_CLASS_
4
5 class PowerUp {
6 private:
7     /* data */
8 public:
9     PowerUp(/* args */) { }
10    ~PowerUp() { }
11 };
12 #endif
```

## 8.49 src/Combat/projectile.cpp File Reference

```
#include "Combat/Projectile.hpp"
```

## 8.50 src/Combat/Projectile.hpp File Reference

```
#include "entity.hpp"
```

## Classes

- class [Projectile](#)

## Macros

- `#define` [\\_Projectile\\_CLASS\\_](#)

## Typedefs

- typedef std::unique\_ptr< [Projectile](#) > [ProjectileUP](#)

### 8.50.1 Macro Definition Documentation

#### 8.50.1.1 \_Projectile\_CLASS\_

```
#define _Projectile_CLASS_
```

### 8.50.2 Typedef Documentation

#### 8.50.2.1 ProjectileUP

```
typedef std::unique_ptr<Projectile> ProjectileUP
```

## 8.51 Projectile.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _Projectile_CLASS_
4 #define _Projectile_CLASS_
5
6 #include "entity.hpp"
7
8 class Character;
9
10 class Projectile : public Entity {
11 public:
12     enum Type { PlayerProjectile,
13                EnemyProjectile };
14
15 public:
16     Projectile(sf::Sprite& sprite, sf::Vector2f startPos, bool penetratesObjects = false);
17
18     bool HasHit(Character* c);
19
20     void Hit(Character* c);
21
22     int GetDamage() { return damage_; }
23
24     Projectile::Type GetType() { return type_; }
25
26     sf::Vector2f GetDirection() { return direction_; }
27
28     sf::Vector2f GetStartPosition() { return startPos_; }
29
30     float GetTimeExisted() { return timeExisted_; }
31
32     float GetTimeLifeSpan() { return timeLifeSpan_; }
33
34     float GetProjectileSpeed() { return projectileSpeed_; }
35
36     float GetDistanceLifeSpan() { return distanceLifeSpanSquared_; }
37
38     bool Penetrates() { return penetrates_; }
```

```

97
104     bool IsAlive() { return alive_; }
105
111     void SetType(Projectile::Type type);
112
118     void SetDirection(sf::Vector2f direction);
119
125     void SetDamage(int damage);
126
132     void SetProjectileSpeed(float projectileSpeed);
133
139     void SetTimeLifeSpan(float timeLifeSpan);
140
146     void SetDistanceLifeSpan(float distanceLifeSpan);
147
153     void SetSprite(sf::Sprite sprite) { sprite_ = sprite; }
154
159     void Kill() { alive_ = false; };
160
166     void Update(float dt);
167
168 private:
169     Type type_;
170     sf::Vector2f direction_;
171     sf::Vector2f startPos_;
172     std::set<Character*> charactersHit_;
173     int damage_;
174     float projectileSpeed_;
175
176     bool alive_;
177     bool penetrates_;
178     float distanceLifeSpanSquared_;
179     float timeExisted_;
180     float timeLifeSpan_;
181
182     void initVariables();
183     bool move(float dt, float x, float y);
184 };
185
186 typedef std::unique_ptr<Projectile> ProjectileUP;
187
188 #endif

```

## 8.52 src/Combat/Weapons/BowWeapon.cpp File Reference

```
#include "Combat/Weapons/BowWeapon.hpp"
```

## 8.53 src/Combat/Weapons/BowWeapon.hpp File Reference

```
#include "Combat/Weapons/Weapon.hpp"
```

### Classes

- class [BowWeapon](#)

### Macros

- #define [\\_BOWWEAPON\\_CLASS\\_](#)

#### 8.53.1 Macro Definition Documentation



8.53.1.1 `_BOWWEAPON_CLASS_`

```
#define _BOWWEAPON_CLASS_
```

## 8.54 BowWeapon.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _BOWWEAPON_CLASS_
3 #define _BOWWEAPON_CLASS_
4
5 #include "Combat/Weapons/Weapon.hpp"
6
7 class BowWeapon : public Weapon {
8 private:
9 public:
10     BowWeapon(int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize,
11               const std::string& spriteLocation);
12     virtual ~BowWeapon() { }
13     virtual ProjectileUP Use(Vector2f dir, Vector2f origin);
14 };
15 #endif
```

## 8.55 src/Combat/Weapons/SwordWeapon.cpp File Reference

```
#include "Combat/Weapons/SwordWeapon.hpp"
```

## 8.56 src/Combat/Weapons/SwordWeapon.hpp File Reference

```
#include "Combat/Weapons/Weapon.hpp"
```

## Classes

- class [SwordWeapon](#)

## Macros

- #define [\\_SWORDWEAPON\\_CLASS\\_](#)

## 8.56.1 Macro Definition Documentation

8.56.1.1 `_SWORDWEAPON_CLASS_`

```
#define _SWORDWEAPON_CLASS_
```

## 8.57 SwordWeapon.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _SWORDWEAPON_CLASS_
3 #define _SWORDWEAPON_CLASS_
4
5 #include "Combat/Weapons/Weapon.hpp"
6
7 class SwordWeapon : public Weapon {
8 private:
9 public:
10     SwordWeapon(int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize,
11                 const std::string& spriteLocation);
12     virtual ~SwordWeapon() { }
13     virtual ProjectileUP Use(Vector2f dir, Vector2f origin);
14 };
15 #endif
```

## 8.58 src/Combat/Weapons/Weapon.cpp File Reference

```
#include "Combat/Weapons/Weapon.hpp"
```

## 8.59 src/Combat/Weapons/Weapon.hpp File Reference

```
#include "Combat/PowerUps/PowerUp.hpp"
#include "Combat/Projectile.hpp"
#include "Utility/SpriteHelper.hpp"
```

### Classes

- class [Weapon](#)

### Macros

- #define [\\_WEAPON\\_CLASS\\_](#)

### 8.59.1 Macro Definition Documentation

#### 8.59.1.1 \_WEAPON\_CLASS\_

```
#define _WEAPON_CLASS_
```

## 8.60 Weapon.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #ifndef _WEAPON_CLASS_
3 #define _WEAPON_CLASS_
4
5 #include "Combat/PowerUps/PowerUp.hpp"
6 #include "Combat/Projectile.hpp"
7 #include "Utility/SpriteHelper.hpp"
8
9 using namespace std;
10 using namespace sf;
11
12 class Weapon {
13 public:
14     Weapon(int damage, int range, int rateOfFire, float projectileSpeed, Vector2f projectileSize, const
        std::string& spriteLocation);
15     virtual ~Weapon() {};
16     virtual ProjectileUP Use(Vector2f dir, Vector2f origin) = 0;
17     // virtual void Animate() = 0;
18     void AddPowerUp(PowerUp* up);
19     float GetAttackCooldown() { return cooldown_; };
20     float GetRange() { return range_; };
21     void SetTextureRect(sf::IntRect rect);
22     void BoostDamageValue();
23     void UnBoostDamageValue();
24
25 protected:
26     int defaultDamage_;
27     int currentDamage_;
28     float damageBoostModifier = 1.1f;
29
30     int range_;
31     float projectileSpeed_;
32     Vector2f projectileSize_;
33     int speed_;
34     bool penetrates_ = false;
35     const int maxPowerUps = 3;
36     int getPowerUpCount();
37     Sprite sprite_;
38     Texture texture_;
39     vector<PowerUp*> powerUps_;
40     float cooldown_;
41     float attackCooldownLength_;
42     float attackCooldownLeft;
43 };
44 #endif

```

## 8.61 src/Dungeon/map.cpp File Reference

```

#include "map.hpp"
#include <time.h>

```

## 8.62 src/Dungeon/map.hpp File Reference

```

#include "Actors/player.hpp"
#include "Dungeon/specialrooms/BossRoom.hpp"
#include "Dungeon/specialrooms/TreasureRoom.hpp"
#include "Dungeon/specialrooms/startingRoom.hpp"
#include "PCH.hpp"
#include "roomInstance.hpp"

```

### Classes

- class [Map](#)

## Macros

- `#define _MAP_`

### 8.62.1 Macro Definition Documentation

#### 8.62.1.1 \_MAP\_

```
#define _MAP_
```

## 8.63 map.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _MAP_
3 #define _MAP_
4 #include "Actors/player.hpp"
5 #include "Dungeon/specialrooms/BossRoom.hpp"
6 #include "Dungeon/specialrooms/TreasureRoom.hpp"
7 #include "Dungeon/specialrooms/startingRoom.hpp"
8 #include "PCH.hpp"
9 #include "roomInstance.hpp"
10
11 class Map {
12 public:
13     Map(sf::Vector2u sizeOfRooms, int noRooms, PlayerPS player);
14     ~Map();
15
16     void RenderCurrentRoom(sf::RenderTarget* target);
17
18     void CreateDungeon(int noRooms);
19
20     void MovePlayer(Direction dir);
21
22     RoomInstance* GetRoomInDir(Direction direction);
23
24     sf::Vector2i DirToVec(Direction direction);
25
26     RoomInstance* GetCurrentRoom();
27
28     RoomInstance* GetSpawnRoom();
29
30     void ResetMap();
31
32     bool IsBossRoomCleared();
33 private:
34     std::pair<int, int> findBossRoom(std::map<std::pair<int, int>, std::set<Direction> coordsAndWalls>);
35     void Move(Direction dir);
36
37     RoomInstance* GetRoomAt(sf::Vector2i coord);
38
39     std::pair<int, int> getKey();
40
41     std::pair<int, int> getKey(sf::Vector2i coord);
42     RoomInstance* getRandomRoom();
43     RoomInstance* addRoomToDungeon(sf::Vector2u roomSize, sf::Vector2i coords);
44     void addStartingRoomToDungeon(sf::Vector2u roomSize, sf::Vector2i coords);
45     sf::Vector2u roomSize_;
46     sf::Vector2i currentPos_;
47     PlayerPS player_;
48     sf::Vector2i spawnCoords_;
49     sf::Vector2i bossCoords_;
50     std::map<std::pair<int, int>, RoomInstance*> dungeon_; // cant use vector2i as a key
51     std::vector<std::pair<int, int> existingRoomCoords_;
```

## 8.64 src/Dungeon/roomInstance.cpp File Reference

```
#include "roomInstance.hpp"
#include <time.h>
```

### Namespaces

- namespace [direction](#)

### Functions

- [Direction direction::GetOppositeDir](#) ([Direction](#) direction)

## 8.65 src/Dungeon/roomInstance.hpp File Reference

```
#include "Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp"
#include "Combat/Health/Potion.hpp"
#include "Tiles/roomTile.hpp"
```

### Classes

- class [RoomInstance](#)  
*Class representing a room of a dungeon, usually includes a monsterspawner.*

### Namespaces

- namespace [direction](#)

### Enumerations

- enum class [Direction](#) {  
    [Up](#) , [Down](#) , [Left](#) , [Right](#) ,  
    [Count](#) }

### Functions

- [Direction direction::GetOppositeDir](#) ([Direction](#) direction)

### 8.65.1 Enumeration Type Documentation

#### 8.65.1.1 Direction

```
enum class Direction [strong]
```

## Enumerator

Up	
Down	
Left	
Right	
Count	

## 8.66 roomInstance.hpp

[Go to the documentation of this file.](#)

```

1
2 #ifndef _ROOM_INSTANCE_
3 #define _ROOM_INSTANCE_
4
5 #include "Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp"
6 #include "Combat/Health/Potion.hpp"
7 #include "Tiles/roomTile.hpp"
8
9 enum class Direction {
10     Up,
11     Down,
12     Left,
13     Right,
14     Count
15 };
16
17 namespace direction {
18 Direction GetOppositeDir(Direction direction);
19 }
20
21 class RoomInstance {
22 public:
23     RoomInstance(sf::Vector2u window_size, sf::Vector2i coords);
24     RoomInstance(sf::Vector2u window_size, sf::Vector2i coords, MonsterSpawner* spawner);
25     RoomInstance() = default;
26     virtual ~RoomInstance();
27
28     void Render(sf::RenderTarget* target);
29
30     std::vector<RoomTile> getRoomTilesAt(sf::FloatRect bounds);
31
32     bool positionIsWalkable(sf::FloatRect bounds);
33     bool positionIsPenetratable(sf::FloatRect bounds);
34
35     std::vector<std::vector<RoomTile>> getTiles() const;
36
37     void CreateExit(Direction direction);
38
39     void Connect(Direction dir, RoomInstance* room);
40
41     RoomInstance* GetRoomInDir(Direction dir);
42
43     sf::Vector2i GetCoords() { return coords_; }
44
45     void renderSpriteBackground();
46
47     sf::Vector2u GetEntranceInDirection(Direction direction);
48
49     virtual void Enter(PlayerPS player, Direction direction);
50
51     void Exit();
52
53     std::vector<MonsterSP>& GetMonsters();
54
55     void deleteMonster(MonsterSP m);
56
57     Direction RemoveRandomDirection();
58
59     void RemoveDirection(Direction dir);
60
61     bool HasDirectionsLeft();
62
63     bool IsCleared();
64
65 };
66
67 #endif

```

```

138     bool IsVisisted() {return visited_;}
139
140     bool monsterCleared();
141
142     void AddPotion(Potion* potion);
143
144     std::vector<Potion*>& GetPotions();
145
146 protected:
151     virtual void setTiles();
152     sf::Vector2u roomSize_;
153     sf::Vector2i coords_;
154     std::vector<std::vector<RoomTile*>> tileVector_;
155     sf::RenderTexture roomTexture;
156     sf::Sprite roomBackground;
157     std::vector<MonsterSP> monsters_;
158     MonsterSpawner* spawner_;
159     std::vector<Potion*> potions_;
160     bool cleared_; // whether the room is cleared
161     bool visited_; // whether the room has been visited already
162
163     vector<Direction> directionsLeft_;
164 };
165
166 #endif

```

## 8.67 src/Dungeon/specialrooms/BossRoom.cpp File Reference

```
#include "BossRoom.hpp"
```

## 8.68 src/Dungeon/specialrooms/BossRoom.hpp File Reference

```

#include "Actors/Monsters/MonsterSpawner/BossSpawner.hpp"
#include "Dungeon/roomInstance.hpp"

```

### Classes

- class [BossRoom](#)

*The roominstance where the boss spawns and which has to be cleared to beat the game.*

## 8.69 BossRoom.hpp

[Go to the documentation of this file.](#)

```

1 #ifndef _BOSS_ROOM_
2 #define _BOSS_ROOM_
3
4 #include "Actors/Monsters/MonsterSpawner/BossSpawner.hpp"
5 #include "Dungeon/roomInstance.hpp"
6
11 class BossRoom : public RoomInstance {
12 public:
13
18     BossRoom(sf::Vector2u window_size, sf::Vector2i coords);
19     BossRoom() { }
20     ~BossRoom() { }
27     virtual void Enter(PlayerPS player, Direction direction);
28     virtual void setTiles(sf::Vector2u window_size);
29 };
30
31 #endif

```

## 8.70 src/Dungeon/specialrooms/startingRoom.cpp File Reference

```
#include "startingRoom.hpp"
```

## 8.71 src/Dungeon/specialrooms/startingRoom.hpp File Reference

```
#include "Dungeon/roomInstance.hpp"
```

### Classes

- class [StartingRoom](#)

### Macros

- `#define` [\\_STARTING\\_ROOM\\_CLASS\\_](#)

### 8.71.1 Macro Definition Documentation

#### 8.71.1.1 \_STARTING\_ROOM\_CLASS\_

```
#define _STARTING_ROOM_CLASS_
```

## 8.72 startingRoom.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _STARTING_ROOM_CLASS_
3 #define _STARTING_ROOM_CLASS_
4 #include "Dungeon/roomInstance.hpp"
5
6 class StartingRoom : public RoomInstance {
7 public:
14     StartingRoom(sf::Vector2u window_size, sf::Vector2i choords);
15     StartingRoom() { }
16     ~StartingRoom() { }
17     virtual void setTiles(sf::Vector2u window_size);
18
19 private:
20 };
21
22 #endif
```

## 8.73 src/Dungeon/specialrooms/TreasureRoom.cpp File Reference

```
#include "TreasureRoom.hpp"
```



## 8.74 src/Dungeon/specialrooms/TreasureRoom.hpp File Reference

```
#include "Dungeon/roomInstance.hpp"
```

### Classes

- class [TreasureRoom](#)  
*Unused, was going to represent a room that includes a treasure and no monsters.*

## 8.75 TreasureRoom.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef _TREASURE_ROOM_
2 #define _TREASURE_ROOM_
3
4 #include "Dungeon/roomInstance.hpp"
5
10 class TreasureRoom : public RoomInstance {
11 public:
12     TreasureRoom(sf::Vector2u window_size, sf::Vector2i choords);
13     TreasureRoom() { }
14     ~TreasureRoom() { }
15     virtual void setTiles(sf::Vector2u window_size);
16
17 private:
18 };
19
20 #endif
```

## 8.76 src/Dungeon/Tiles/roomTile.cpp File Reference

```
#include "roomTile.hpp"
```

## 8.77 src/Dungeon/Tiles/roomTile.hpp File Reference

```
#include "PCH.hpp"
```

### Classes

- class [RoomTile](#)
- class [WallTile](#)
- class [FrontWallTile](#)  
*A wall that is not walkable but penetrable.*
- class [FloorTile](#)

## 8.78 roomTile.hpp

[Go to the documentation of this file.](#)

```

1  #ifndef _ROOM_TILE_
2  #define _ROOM_TILE_
3
4  #include "PCH.hpp"
5
6  class RoomTile {
7  public:
8      RoomTile(std::string texture, float x, float y, bool walkable, bool penetratable);
9
10     const sf::Vector2f getSize() const;
11     const sf::Vector2f& getPosition() const;
12     const sf::FloatRect getBoundingBox() const;
13     const sf::Sprite& getSprite() const;
14     bool isWalkable() const;
15     bool isPenetratable() const;
16
17 private:
18     sf::Vector2f position;
19     sf::Texture tileTexture;
20     sf::Sprite tileSprite;
21     bool walkable_;
22     bool penetratable_;
23
24     bool setTileTexture(std::string);
25 };
26
27 class WallTile : public RoomTile {
28 public:
29     WallTile(std::string texture, float x, float y)
30         : RoomTile(texture, x, y, false, false)
31     {
32     }
33 };
34
35 class FrontWallTile : public RoomTile {
36 public:
37     FrontWallTile(std::string texture, float x, float y)
38         : RoomTile(texture, x, y, false, true)
39     {
40     }
41 };
42
43 class FloorTile : public RoomTile {
44 public:
45     FloorTile(std::string texture, float x, float y)
46         : RoomTile(texture, x, y, true, true)
47     {
48     }
49 };
50 #endif

```

## 8.79 src/entity.cpp File Reference

```
#include "entity.hpp"
```

## 8.80 src/entity.hpp File Reference

### Classes

- class [Entity](#)

### Macros

- #define [\\_ENTITY\\_CLASS\\_](#)

## 8.80.1 Macro Definition Documentation

### 8.80.1.1 `_ENTITY_CLASS_`

```
#define _ENTITY_CLASS_
```

## 8.81 entity.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _ENTITY_CLASS_
4 #define _ENTITY_CLASS_
5 class Entity {
6 public:
15     Entity(const std::string& spriteLocation, float xPos, float yPos, sf::Vector2f spriteDims);
16     Entity(const std::string& spriteLocation, sf::Vector2f pos, sf::Vector2f spriteDims);
17     Entity(sf::Sprite& sprite, float xPos, float yPos);
18     Entity(sf::Sprite& sprite, sf::Vector2f pos);
19
20     virtual ~Entity() {};
21
27     const sf::Sprite& GetSprite() const { return sprite_; };
28
34     const sf::Vector2f& GetPos() const { return pos_; }
35
41     const sf::Vector2i GetPosI() { return sf::Vector2i(pos_); }
42
48     sf::Vector2f GetSpritePosition() const { return sprite_.getPosition(); } // might be unnecessary,
    because sprite pos should be same as that returned of GetPos()
49
55     sf::Vector2f GetSpriteCenter() const;
56
62     const sf::Vector2f& GetOldPosition() const;
63
64     sf::FloatRect GetSpriteBounds() const;
65     sf::FloatRect GetBaseBoxAt(sf::Vector2f pos) const;
66
72     void SetPos(sf::Vector2f pos);
73
79     void SetPosAndOldPos(sf::Vector2f pos);
80
81     virtual void Render(sf::RenderTarget* target);
82
83 protected:
84     void initSprite(const std::string& spriteLocation, sf::Vector2f spriteDims);
85
86     sf::Vector2f pos_;
87     sf::Vector2f oldPos_;
88     sf::Sprite sprite_;
89     sf::Texture texture_;
90 };
91 #endif
```

## 8.82 src/game.cpp File Reference

```
#include "game.hpp"
#include "Utility/Collision.hpp"
```

### Macros

- `#define C_PIXELS 64`

## 8.82.1 Macro Definition Documentation

### 8.82.1.1 C\_PIXELS

```
#define C_PIXELS 64
```

## 8.83 src/game.hpp File Reference

```
#include "Actors/Monsters/BossMonster.hpp"  
#include "Actors/player.hpp"  
#include "Combat/Health/Potion.hpp"  
#include "Combat/Projectile.hpp"  
#include "Combat/Weapons/BowWeapon.hpp"  
#include "Combat/Weapons/SwordWeapon.hpp"  
#include "Dungeon/map.hpp"  
#include "Utility/LevelUpSystem.hpp"  
#include "Utility/ScreenText.hpp"  
#include "Utility/Sounds/SoundEffects.hpp"  
#include "gamebar.hpp"
```

### Classes

- class [Game](#)

### Macros

- #define [\\_GAME\\_CLASS\\_](#)

## 8.83.1 Macro Definition Documentation

### 8.83.1.1 \_GAME\_CLASS\_

```
#define _GAME_CLASS_
```

## 8.84 game.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2
3 #ifndef _GAME_CLASS_
4 #define _GAME_CLASS_
5
6 #include "Actors/Monsters/BossMonster.hpp"
7 #include "Actors/player.hpp"
8 #include "Combat/Health/Potion.hpp"
9 #include "Combat/Projectile.hpp"
10 #include "Combat/Weapons/BowWeapon.hpp"
11 #include "Combat/Weapons/SwordWeapon.hpp"
12 #include "Dungeon/map.hpp"
13 #include "Utility/LevelUpSystem.hpp"
14 #include "Utility/ScreenText.hpp"
15 #include "Utility/Sounds/SoundEffects.hpp"
16 #include "gamebar.hpp"
17
18 class Game {
19 public:
20     Game();
21     ~Game();
22
23     void UpdateGame();
24     void RenderGame();
25     bool Running() const;
26     void Events();
27
28 private:
29     sf::VideoMode videomode_;
30     sf::RenderWindow* window_;
31     sf::Event event_;
32     sf::Clock dtClock_;
33     PlayerPS player_;
34     Map dungeonMap_;
35     Gamebar gamebar_;
36     ScreenText deathText_;
37     ScreenText victoryScreen_;
38
39     SoundEffect* playerHitSound = new SoundEffect("content/sounds/playerHit.wav");
40     SoundEffect* monsterHitSound = new SoundEffect("content/sounds/monsterHit.wav");
41
42     float dt_;
43     bool paused = false;
44     bool escapePressedLastTick = paused;
45     bool gameEnder_;
46
47     std::list<ProjectileUP> projectiles_;
48
49     void initWindow();
50     void updateDt();
51     void manageInput();
52     void managePauseInput();
53     void checkCollisions(Character* character, Projectile::Type projectileType);
54     void checkMonsterCollisions();
55     void checkPlayerCollisions();
56     void checkAndHandleProjectileWallCollisions();
57     void deleteProjectile(ProjectileUP p);
58     void addProjectiles(std::list<ProjectileUP> listToAdd);
59     void deleteMonster(Character* m);
60     void deletePotion(Potion* p);
61     void updateProjectiles();
62     void updateMonsters();
63     void updatePotions();
64     bool collidesWithWall(Character* character);
65
66     bool collidesWithWall(Projectile* object);
67     bool ShouldChangeRoom();
68     bool gameLost();
69     void restartGame();
70
71     bool gameWon();
72 };
73
74 #endif

```

## 8.85 src/gamebar.cpp File Reference

```
#include "gamebar.hpp"
```

## 8.86 src/gamebar.hpp File Reference

```
#include "Actors/player.hpp"
#include "Combat/Health/HealthPotions.hpp"
#include "Combat/Health/Potion.hpp"
#include "Utility/SpriteHelper.hpp"
```

### Classes

- class [Gamebar](#)

### Macros

- #define [\\_GAMEBAR\\_CLASS\\_](#)

### 8.86.1 Macro Definition Documentation

#### 8.86.1.1 [\\_GAMEBAR\\_CLASS\\_](#)

```
#define _GAMEBAR_CLASS_
```

## 8.87 gamebar.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _GAMEBAR_CLASS_
4 #define _GAMEBAR_CLASS_
5
6 #include "Actors/player.hpp"
7 #include "Combat/Health/HealthPotions.hpp"
8 #include "Combat/Health/Potion.hpp"
9 #include "Utility/SpriteHelper.hpp"
10 class Gamebar {
11 public:
12     Gamebar(PlayerPS player);
13
14     Gamebar() {};
15     void Render(sf::RenderTarget* target);
16     void Update();
17
18     void RenderInventory(sf::RenderTarget* target);
19
20 private:
21     PlayerPS player_;
22     sf::Font font_;
23     sf::RectangleShape background_;
24     sf::RectangleShape greenBar_;
25     sf::RectangleShape redBar_;
26     sf::RectangleShape violetBar_;
27     sf::RectangleShape yellowBar_;
28     sf::Text hp_;
29
30     sf::Vector2f redpos_ = sf::Vector2f(800, 5);
31     sf::Vector2f greenpos_ = sf::Vector2f(900, 5);
32     sf::Vector2f yellowpos_ = sf::Vector2f(1000, 5);
```

```

47     sf::Vector2f violetpos_ = sf::Vector2f(1100, 5);
48
49     sf::Sprite redsprite_;
50     sf::Texture redtexture_;
51
52     sf::Sprite greensprite_;
53     sf::Texture greentexture_;
54
55     sf::Sprite yellowsprite_;
56     sf::Texture yellowtexture_;
57
58     sf::Sprite violetsprite_;
59     sf::Texture violetttexture_;
60 };
61 #endif

```

## 8.88 src/Interfaces/CollisionSystem.hpp File Reference

```
#include "Interfaces/ICollidable.hpp"
```

### Classes

- class [CollisionSystem](#)  
*Unused, was not intergrated or completed.*

### Macros

- #define [\\_COLLISIONSYSTEM\\_CLASS\\_](#)

### 8.88.1 Macro Definition Documentation

#### 8.88.1.1 \_COLLISIONSYSTEM\_CLASS\_

```
#define _COLLISIONSYSTEM_CLASS_
```

## 8.89 CollisionSystem.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #ifndef _COLLISIONSYSTEM_CLASS_
3 #define _COLLISIONSYSTEM_CLASS_
4 #include "Interfaces/ICollidable.hpp"
5
10 class CollisionSystem {
11 private:
12     // hold all objects that participate in collision
13     std::vector<ICollidable*> objectList;
14
15 public:
16     CollisionSystem() {};
17     void AddToCollisionList(ICollidable* obj);
18     void RemoveFromCollisionList(ICollidable* obj);
19     void ProcessCollisionList();
20 };
21
22 #endif

```

## 8.90 src/Interfaces/ICollidable.hpp File Reference

### Classes

- class [ICollidable](#)  
*Unused interface for collisionsystem.*

### Macros

- `#define` [\\_COLLIDABLE\\_INTERFACE\\_](#)

### 8.90.1 Macro Definition Documentation

#### 8.90.1.1 \_COLLIDABLE\_INTERFACE\_

```
#define _COLLIDABLE_INTERFACE_
```

## 8.91 ICollidable.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _COLLIDABLE_INTERFACE_
3 #define _COLLIDABLE_INTERFACE_
4
9 class ICollidable {
10 public:
11     enum EntityType {
12         character,
13         projectile,
14         tile,
15     };
16
17     // we use these function to retrieve collision relevant information
18     // virtual sf::Vector2f GetPosition() = 0; // objects have position
19     virtual sf::FloatRect GetBoundingBox() = 0; // objects have a size
20
21     // using this function, we notify the object that it collided with something else
22     virtual void ProcessCollision(ICollidable* other) = 0;
23
24     virtual EntityType GetEntityType() = 0;
25
26     virtual ~ICollidable() {};
27 };
28
29 #endif
```



## 8.92 src/PCH.hpp File Reference

```
#include <SFML/Audio.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Network.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <algorithm>
#include <cstdint>
#include <deque>
#include <fstream>
#include <iostream>
#include <list>
#include <map>
#include <memory>
#include <set>
#include <string>
#include <vector>
#include <atomic>
#include <cassert>
#include <cmath>
#include <cstdlib>
#include <exception>
#include <functional>
#include <iomanip>
#include <mutex>
#include <random>
#include <sstream>
#include <thread>
#include <type_traits>
#include "Utility/FileSystem.hpp"
#include "Utility/Types.hpp"
```

### Macros

- `#define NDEBUG`
- `#define UNUSED(x) (void)(x)`

### 8.92.1 Macro Definition Documentation

#### 8.92.1.1 NDEBUG

```
#define NDEBUG
```

#### 8.92.1.2 UNUSED

```
#define UNUSED(  
    x ) (void)(x)
```

## 8.93 PCH.hpp

[Go to the documentation of this file.](#)

```

1 #ifndef PRECOMPILED_HEADER_HPP
2 #define PRECOMPILED_HEADER_HPP
3
4 #ifndef _DEBUG
5 #ifndef NDEBUG
6 #define NDEBUG
7 #endif
8 #endif // _DEBUG
9
10 // SFML
11 #include <SFML/Audio.hpp>
12 #include <SFML/Graphics.hpp>
13 #include <SFML/Network.hpp>
14 #include <SFML/System.hpp>
15 #include <SFML/Window.hpp>
16
17 // Raspberry Pi
18 #ifdef SFML_SYSTEM_LINUX
19 #ifdef __arm__
20 #define SFML_SYSTEM_PI
21 #endif
22 #endif // SFML_SYSTEM_LINUX
23
24 // Typical stdafx.h
25 #include <algorithm>
26 #include <cstdio>
27 #include <deque>
28 #include <fstream>
29 #include <iostream>
30 #include <list>
31 #include <map>
32 #include <memory>
33 #include <set>
34 #include <string>
35 #include <vector>
36
37 // Additional C/C++ libs
38 #include <atomic>
39 #include <cassert>
40 #include <cmath>
41 #include <cstdlib>
42 #include <exception>
43 #include <functional>
44 #include <iomanip>
45 #include <mutex>
46 #include <random>
47 #include <sstream>
48 #include <thread>
49 #include <type_traits>
50
51 // Windows
52 #ifdef _WIN32
53 #ifndef UNICODE
54 #define UNICODE
55 #endif
56
57 #ifndef _UNICODE
58 #define _UNICODE
59 #endif
60
61 #define WIN32_LEAN_AND_MEAN
62 // #include <windows.h>
63 #endif // _WIN32
64
65 // Utils
66 #include "Utility/FileSystem.hpp"
67 #include "Utility/Types.hpp"
68
69 // Macros
70 #define UNUSED(x) (void)(x)
71
72 #endif // PRECOMPILED_HEADER_HPP

```

## 8.94 src/Platform/IPlatform.hpp File Reference

### Classes

- struct [util::IPlatform](#)

## Namespaces

- namespace `util`

## 8.95 IPlatform.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef UTIL_IPLATFORM_HELPER_HPP
2 #define UTIL_IPLATFORM_HELPER_HPP
3
4 namespace util
5 {
6     struct IPlatform
7     {
8         virtual ~IPlatform() = default;
9         virtual void setIcon(const sf::WindowHandle &inHandle) = 0;
10        virtual void toggleFullscreen(const sf::WindowHandle &inHandle, const sf::Uint32 inStyle, const
            bool inWindowed, const sf::Vector2u &inResolution) = 0;
11        virtual int getRefreshRate(const sf::WindowHandle &inHandle) = 0;
12        virtual float getScreenScalingFactor(const sf::WindowHandle &inHandle) = 0;
13    };
14 }
15
16 #endif // UTIL_IPLATFORM_HELPER_HPP
```

## 8.96 src/Platform/Platform.hpp File Reference

### Namespaces

- namespace `util`

## 8.97 Platform.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef UTIL_PLATFORM_HPP
2 #define UTIL_PLATFORM_HPP
3
4 #if defined(__APPLE__)
5     #include "Platform/MacOS/MacOSPlatform.hpp"
6 #elif defined(__linux__)
7     #include "Platform/Unix/LinuxPlatform.hpp"
8 #elif defined(_WIN32)
9     #include "Platform/Win32/WindowsPlatform.hpp"
10 #endif
11
12 namespace util
13 {
14     #if defined(__APPLE__)
15     using Platform = MacOSPlatform;
16     #elif defined(__linux__)
17     using Platform = LinuxPlatform;
18     #elif defined(_WIN32)
19     using Platform = WindowsPlatform;
20 #endif
21 }
22 #endif // UTIL_PLATFORM_HPP
```

## 8.98 src/Platform/Unix/LinuxPlatform.cpp File Reference

## 8.99 src/Platform/Unix/LinuxPlatform.hpp File Reference

```
#include "Platform/IPlatform.hpp"
```

## Classes

- struct [util::LinuxPlatform](#)

## Namespaces

- namespace [util](#)

## 8.100 LinuxPlatform.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef UTIL_LINUX_PLATFORM_HPP
2 #define UTIL_LINUX_PLATFORM_HPP
3
4 #include "Platform/IPlatform.hpp"
5
6 namespace util
7 {
8     struct LinuxPlatform : IPlatform
9     {
10         LinuxPlatform();
11
12         void setIcon(const sf::WindowHandle& inHandle) final;
13         void toggleFullscreen(const sf::WindowHandle& inHandle, const sf::Uint32 inStyle, const bool
            inWindowed, const sf::Vector2u& inResolution) final;
14         float getScreenScalingFactor(const sf::WindowHandle& inHandle) final;
15         int getRefreshRate(const sf::WindowHandle& inHandle) final;
16     };
17 }
18
19 #endif // UTIL_LINUX_PLATFORM_HPP
```

## 8.101 src/Platform/Win32/Resource.h File Reference

```
#include <winuser.h>
```

## Macros

- #define [WIN32\\_ICON\\_MAIN](#) 1
- #define [MANIFEST\\_RESOURCE\\_ID](#) 1

### 8.101.1 Macro Definition Documentation

#### 8.101.1.1 MANIFEST\_RESOURCE\_ID

```
#define MANIFEST_RESOURCE_ID 1
```

## 8.101.1.2 WIN32\_ICON\_MAIN

```
#define WIN32_ICON_MAIN 1
```

## 8.102 Resource.h

[Go to the documentation of this file.](#)

```
1 #ifndef RESOURCE_H
2 #define RESOURCE_H
3
4 #include <winuser.h>
5
6 #define WIN32_ICON_MAIN 1
7 #define MANIFEST_RESOURCE_ID 1
8
9 #endif // RESOURCE_H
```

## 8.103 src/Platform/Win32/WindowsPlatform.cpp File Reference

## 8.104 src/Platform/Win32/WindowsPlatform.hpp File Reference

```
#include "Platform/IPlatform.hpp"
```

## Classes

- struct [util::WindowsPlatform](#)

## Namespaces

- namespace [util](#)

## 8.105 WindowsPlatform.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef UTIL_WINDOWS_PLATFORM_HPP
2 #define UTIL_WINDOWS_PLATFORM_HPP
3
4 #include "Platform/IPlatform.hpp"
5
6 // TODO: WM_DISPLAYCHANGE event handling (multi-monitor support)
7
8 namespace util
9 {
10     struct WindowsPlatform : IPlatform
11     {
12         WindowsPlatform();
13
14         void setIcon(const sf::WindowHandle& inHandle) final;
15         void toggleFullscreen(const sf::WindowHandle& inHandle, const sf::Uint32 inStyle, const bool
inWindowed, const sf::Vector2u& inResolution) final;
16         float getScreenScalingFactor(const sf::WindowHandle& inHandle) final;
17         int getRefreshRate(const sf::WindowHandle& inHandle) final;
18
19     private:
20         PBYTE getIconDirectory(const int inResourceId);
21         HICON getIconFromIconDirectory(PBYTE inIconDirectory, const uint inSize);
22         DWORD sFmlWindowStyleToWin32WindowStyle(const sf::Uint32 inStyle);
23
24         float m_screenScalingFactor = 0.0f;
25         int m_refreshRate = 0;
26     };
27 }
28
29 #endif // UTIL_WINDOWS_PLATFORM_HPP
```

## 8.106 src/readme.md File Reference

## 8.107 src/Utility/Collision.cpp File Reference

```
#include "Collision.hpp"
#include "SFML/Graphics.hpp"
#include <map>
```

### Classes

- class [Collision::BitmaskManager](#)
- class [Collision::OrientedBoundingBox](#)

### Namespaces

- namespace [Collision](#)

### Functions

- bool [Collision::PixelPerfectTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2, sf::Uint8 AlphaLimit)
- bool [Collision::CreateTextureAndBitmask](#) (sf::Texture &LoadInto, const std::string &Filename)
- sf::Vector2f [Collision::GetSpriteCenter](#) (const sf::Sprite &Object)
- sf::Vector2f [Collision::GetSpriteSize](#) (const sf::Sprite &Object)
- bool [Collision::CircleTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2)
- bool [Collision::BoundingBoxTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2)

### Variables

- BitmaskManager [Collision::Bitmasks](#)

## 8.108 src/Utility/Collision.hpp File Reference

### Namespaces

- namespace [Collision](#)

### Functions

- bool [Collision::PixelPerfectTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2, sf::Uint8 AlphaLimit)
- bool [Collision::CreateTextureAndBitmask](#) (sf::Texture &LoadInto, const std::string &Filename)
- bool [Collision::CircleTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2)
- bool [Collision::BoundingBoxTest](#) (const sf::Sprite &Object1, const sf::Sprite &Object2)
- sf::Vector2f [Collision::GetSpriteCenter](#) (const sf::Sprite &Object)
- sf::Vector2f [Collision::GetSpriteSize](#) (const sf::Sprite &Object)

## 8.109 Collision.hpp

[Go to the documentation of this file.](#)

```

1 /*
2  * File:    collision.h
3  * Authors: Nick Koirala (original version), ahnonay (SFML2 compatibility)
4  *
5  * Collision Detection and handling class
6  * For SFML2.
7
8  Notice from the original version:
9
10 (c) 2009 - LittleMonkey Ltd
11
12 This software is provided 'as-is', without any express or
13 implied warranty. In no event will the authors be held
14 liable for any damages arising from the use of this software.
15
16 Permission is granted to anyone to use this software for any purpose,
17 including commercial applications, and to alter it and redistribute
18 it freely, subject to the following restrictions:
19
20 1. The origin of this software must not be misrepresented;
21 you must not claim that you wrote the original software.
22 If you use this software in a product, an acknowledgment
23 in the product documentation would be appreciated but
24 is not required.
25
26 2. Altered source versions must be plainly marked as such,
27 and must not be misrepresented as being the original software.
28
29 3. This notice may not be removed or altered from any
30 source distribution.
31
32 *
33 * Created on 30 January 2009, 11:02
34 */
35
36 #ifndef COLLISION_H
37 #define COLLISION_H
38
39 namespace Collision {
40 bool PixelPerfectTest(const sf::Sprite& Object1, const sf::Sprite& Object2, sf::Uint8 AlphaLimit = 0);
41
42 bool CreateTextureAndBitmask(sf::Texture& LoadInto, const std::string& Filename);
43
44 bool CircleTest(const sf::Sprite& Object1, const sf::Sprite& Object2);
45
46 bool BoundingBoxTest(const sf::Sprite& Object1, const sf::Sprite& Object2);
47
48 sf::Vector2f GetSpriteCenter(const sf::Sprite& Object);
49 sf::Vector2f GetSpriteSize(const sf::Sprite& Object);
50 }
51
52 #endif /* COLLISION_H */

```

## 8.110 src/Utility/FileSystem.hpp File Reference

```
#include <experimental/filesystem>
```

### Namespaces

- namespace [util](#)

## 8.111 FileSystem.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef UTIL_FILE_SYSTEM_HPP
2 #define UTIL_FILE_SYSTEM_HPP
3
4 // std::filesystem
5 #if __GNUC__ >= 8 || __clang_major__ >= 9
6     #include <filesystem>
7 #else
8     #include <experimental/filesystem>
9 #endif
10
11 namespace util
12 {
13     #if __GNUC__ >= 8 || __clang_major__ >= 9
14     namespace fs = std::filesystem;
15     #else
16     namespace fs = std::experimental::filesystem::v1;
17     #endif
18 }
19
20 #endif // UTIL_FILE_SYSTEM_HPP
```

## 8.112 src/Utility/LevelUpInstance.cpp File Reference

```
#include "LevelUpInstance.hpp"
```

## 8.113 src/Utility/LevelUpInstance.hpp File Reference

### Classes

- class [LevelUpInstance](#)

### Macros

- #define [\\_LevelUpInstance\\_CLASS\\_](#)

### 8.113.1 Macro Definition Documentation

#### 8.113.1.1 [\\_LevelUpInstance\\_CLASS\\_](#)

```
#define _LevelUpInstance_CLASS_
```



## 8.114 LevelUpInstance.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #ifndef _LevelUpInstance_CLASS_
4 #define _LevelUpInstance_CLASS_
5
6 class LevelUpInstance {
7 public:
12     LevelUpInstance();
13
19     void GainXP(float amount);
20
25     void LevelUp();
26
32     int GetLevel() { return level; }
33
39     float GetHPModifier();
40
41 private:
42     int level;
43     float xp;
44     float xpNeededForLevelUp;
45 };
46
47 #endif
```

## 8.115 src/Utility/LevelUpSystem.cpp File Reference

```
#include "LevelUpSystem.hpp"
```

## 8.116 src/Utility/LevelUpSystem.hpp File Reference

```
#include "Actors/character.hpp"
#include "LevelUpInstance.hpp"
```

### Classes

- class [LevelUpSystem](#)

### Macros

- #define [\\_LevelUpSystem\\_CLASS\\_](#)

### 8.116.1 Macro Definition Documentation

#### 8.116.1.1 \_LevelUpSystem\_CLASS\_

```
#define _LevelUpSystem_CLASS_
```

## 8.117 LevelUpSystem.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2
3 #include "Actors/character.hpp"
4 #include "LevelUpInstance.hpp"
5
6 #ifndef _LevelUpSystem_CLASS_
7 #define _LevelUpSystem_CLASS_
8
9 class LevelUpSystem {
10 public:
16     static void AddCharacter(Character* character);
17
24     static void GainXP(Character* character, float amount);
25
31     static void LevelUp(Character* character);
32
39     static int GetLevel(Character* character);
40
47     static float GetHPModifier(Character* character);
48
49     static std::unordered_map<Character*, LevelUpInstance> characterLevelMap;
50 };
51
52 #endif
```

## 8.118 src/Utility/Main.cpp File Reference

```
#include "Platform/Platform.hpp"
#include "game.hpp"
```

### Functions

- int [main](#) ()

#### 8.118.1 Function Documentation

##### 8.118.1.1 main()

```
int main ( )
```

## 8.119 src/Utility/RandomHelper.cpp File Reference

```
#include "Utility/RandomHelper.hpp"
```

### Namespaces

- namespace [randomhelper](#)

## Functions

- float [randomhelper::RandomFloatBetween](#) (float min, float max)
- int [randomhelper::RandomIntBetween](#) (int min, int max)

## 8.120 src/Utility/RandomHelper.hpp File Reference

### Namespaces

- namespace [randomhelper](#)

### Macros

- `#define` [\\_RANDOMHELPER\\_](#)

## Functions

- float [randomhelper::RandomFloatBetween](#) (float min, float max)
- int [randomhelper::RandomIntBetween](#) (int min, int max)

### 8.120.1 Macro Definition Documentation

#### 8.120.1.1 [\\_RANDOMHELPER\\_](#)

```
#define _RANDOMHELPER_
```

## 8.121 RandomHelper.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _RANDOMHELPER_
3 #define _RANDOMHELPER_
4
5 namespace randomhelper {
6     float RandomFloatBetween(float min, float max); //
7     int RandomIntBetween(int min, int max); //Inclusive min and max
8
9 } // namespace
10
11 #endif
```

## 8.122 src/Utility/ScreenText.cpp File Reference

```
#include "ScreenText.hpp"
```

## 8.123 src/Utility/ScreenText.hpp File Reference

```
#include "entity.hpp"
```

### Classes

- class [ScreenText](#)

## 8.124 ScreenText.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef _TEXT_CLASS_
2 #define _TEXT_CLASS_
3
4 #include "entity.hpp"
5
6 class ScreenText : public Entity {
7 public:
8     ScreenText(const std::string& textLocation, sf::Vector2f textPos, sf::Vector2f textDims);
9     ~ScreenText() { }
10
11 private:
12 };
13
14 #endif
```

## 8.125 src/Utility/Sounds/SoundEffects.cpp File Reference

```
#include "SoundEffects.hpp"
```

## 8.126 src/Utility/Sounds/SoundEffects.hpp File Reference

### Classes

- class [SoundEffect](#)

## 8.127 SoundEffects.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef _SOUND_EFFECT_
2 #define _SOUND_EFFECT_
3
4 class SoundEffect {
5 public:
6     SoundEffect(const std::string& SoundEffectFilename);
7     ~SoundEffect() { }
8     void PlaySound();
9
10 private:
11     sf::SoundBuffer buffer;
12     sf::Sound effect;
13 };
14
15 #endif
```

## 8.128 src/Utility/SpriteHelper.cpp File Reference

```
#include "Utility/SpriteHelper.hpp"
```

### Namespaces

- namespace [spritehelper](#)

### Macros

- #define [PI](#) 3.14159265

### Functions

- void [spritehelper::CreateSpriteFrom](#) (const std::string &spriteFile, sf::Vector2f dimensions, sf::Sprite &sprite, sf::Texture &texture)
- void [spritehelper::SetScale](#) (sf::Vector2f wantedDimension, sf::Sprite &sprite)
- void [spritehelper::RotateSprite](#) (sf::Vector2f directionOfRotation, sf::Sprite &sprite)
- void [spritehelper::SetOriginBottomCenter](#) (sf::Sprite &sprite)

### 8.128.1 Macro Definition Documentation

#### 8.128.1.1 PI

```
#define PI 3.14159265
```

## 8.129 src/Utility/SpriteHelper.hpp File Reference

### Namespaces

- namespace [spritehelper](#)

### Macros

- #define [\\_SPRITEHELPER\\_](#)

### Functions

- void [spritehelper::CreateSpriteFrom](#) (const std::string &spriteFile, sf::Vector2f dimensions, sf::Sprite &sprite, sf::Texture &texture)
- void [spritehelper::SetScale](#) (sf::Vector2f wantedDimension, sf::Sprite &sprite)
- void [spritehelper::RotateSprite](#) (sf::Vector2f directionOfRotation, sf::Sprite &sprite)
- void [spritehelper::SetOriginBottomCenter](#) (sf::Sprite &sprite)

## 8.129.1 Macro Definition Documentation

### 8.129.1.1 `_SPRITEHELPER_`

```
#define _SPRITEHELPER_
```

## 8.130 SpriteHelper.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef _SPRITEHELPER_
3 #define _SPRITEHELPER_
4
5 namespace spritehelper {
6 void CreateSpriteFrom(const std::string& spriteFile, sf::Vector2f dimensions, sf::Sprite& sprite,
7                      sf::Texture& texture);
8 void SetScale(sf::Vector2f wantedDimension, sf::Sprite& sprite);
9
10 void RotateSprite(sf::Vector2f directionOfRotation, sf::Sprite& sprite);
11 void SetOriginBottomCenter(sf::Sprite& sprite);
12
13 } // namespace
14
15 #endif
```

## 8.131 src/Utility/Types.hpp File Reference

```
#include <cstdint>
```

### Typedefs

- typedef std::uint8\_t `uchar`
- typedef std::uint16\_t `ushort`
- typedef std::uint32\_t `uint`
- typedef std::uint64\_t `ulong`
- typedef std::int64\_t `long`

### 8.131.1 Typedef Documentation

#### 8.131.1.1 `long`

```
typedef std::int64_t long
```

### 8.131.1.2 uchar

```
typedef std::uint8_t uchar
```

### 8.131.1.3 uint

```
typedef std::uint32_t uint
```

### 8.131.1.4 ullong

```
typedef std::uint64_t ullong
```

### 8.131.1.5 ushort

```
typedef std::uint16_t ushort
```

## 8.132 Types.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef UTIL_TYPES_HPP
2 #define UTIL_TYPES_HPP
3
4 #include <stdint>
5
6 typedef std::uint8_t uchar;
7 typedef std::uint16_t ushort;
8 typedef std::uint32_t uint;
9 typedef std::uint64_t ullong;
10
11 typedef std::int64_t llong;
12
13 #endif // UTIL_TYPES_HPP
```





# Index

- [\\_ANIMATIONHANDLER\\_CLASS\\_](#)
  - [Animationhandler.hpp, 140](#)
- [\\_ANIMATION\\_](#)
  - [animation.hpp, 139](#)
- [\\_BOSS\\_MONSTER\\_CLASS\\_](#)
  - [BossMonster.hpp, 126](#)
- [\\_BOSS\\_SPAWNER\\_CLASS\\_](#)
  - [BossSpawner.hpp, 129](#)
- [\\_BOWWEAPON\\_CLASS\\_](#)
  - [BowWeapon.hpp, 146](#)
- [\\_CHARACTER\\_CLASS\\_](#)
  - [character.hpp, 124](#)
- [\\_COLLIDABLE\\_INTERFACE\\_](#)
  - [ICollidable.hpp, 162](#)
- [\\_COLLISIONSYSTEM\\_CLASS\\_](#)
  - [CollisionSystem.hpp, 161](#)
- [\\_ENTITY\\_CLASS\\_](#)
  - [entity.hpp, 157](#)
- [\\_GAMEBAR\\_CLASS\\_](#)
  - [gamebar.hpp, 160](#)
- [\\_GAME\\_CLASS\\_](#)
  - [game.hpp, 158](#)
- [\\_HEALTHPOTIONS\\_CLASS\\_](#)
  - [HealthPotions.hpp, 142](#)
- [\\_LevelUpInstance\\_CLASS\\_](#)
  - [LevelUpInstance.hpp, 170](#)
- [\\_LevelUpSystem\\_CLASS\\_](#)
  - [LevelUpSystem.hpp, 171](#)
- [\\_MAP\\_](#)
  - [map.hpp, 150](#)
- [\\_MONSTERSPAWNER\\_CLASS\\_](#)
  - [MonsterSpawner.hpp, 130](#)
- [\\_MONSTER\\_CLASS\\_](#)
  - [monster.hpp, 127](#)
- [\\_PLAYER\\_CLASS\\_](#)
  - [player.hpp, 137](#)
- [\\_POTION\\_CLASS\\_](#)
  - [Potion.hpp, 143](#)
- [\\_POWERUP\\_CLASS\\_](#)
  - [PowerUp.hpp, 144](#)
- [\\_Projectile\\_CLASS\\_](#)
  - [Projectile.hpp, 145](#)
- [\\_RANDOMHELPER\\_](#)
  - [RandomHelper.hpp, 173](#)
- [\\_RANDOM\\_MONSTER\\_CLASS\\_](#)
  - [RandomMonster.hpp, 132](#)
- [\\_SEARCHING\\_MONSTER\\_CLASS\\_](#)
  - [SearchingMonster.hpp, 133](#)
- [\\_SLOW\\_MONSTER\\_CLASS\\_](#)
  - [SlowMonster.hpp, 134](#)
- [\\_SNIPING\\_MONSTER\\_CLASS\\_](#)
  - [SnipingMonster.hpp, 135](#)
- [\\_SPRITEHELPER\\_](#)
  - [SpriteHelper.hpp, 176](#)
- [\\_STARTING\\_ROOM\\_CLASS\\_](#)
  - [startingRoom.hpp, 154](#)
- [\\_SWORDWEAPON\\_CLASS\\_](#)
  - [SwordWeapon.hpp, 147](#)
- [\\_WALL\\_PATROL\\_MONSTER\\_CLASS\\_](#)
  - [WallPatrolMonster.hpp, 136](#)
- [\\_WEAPON\\_CLASS\\_](#)
  - [Weapon.hpp, 148](#)
- [~Animation](#)
  - [Animation, 18](#)
- [~AnimationHandler](#)
  - [AnimationHandler, 20](#)
- [~BitmaskManager](#)
  - [Collision::BitmaskManager, 21](#)
- [~BossMonster](#)
  - [BossMonster, 23](#)
- [~BossRoom](#)
  - [BossRoom, 25](#)
- [~BossSpawner](#)
  - [BossSpawner, 27](#)
- [~BowWeapon](#)
  - [BowWeapon, 28](#)
- [~Character](#)
  - [Character, 30](#)
- [~Entity](#)
  - [Entity, 41](#)
- [~Game](#)
  - [Game, 47](#)
- [~ICollidable](#)
  - [ICollidable, 51](#)
- [~IPlatform](#)
  - [util::IPlatform, 52](#)
- [~Map](#)
  - [Map, 60](#)
- [~Monster](#)
  - [Monster, 64](#)
- [~MonsterSpawner](#)
  - [MonsterSpawner, 68](#)
- [~Player](#)
  - [Player, 71](#)
- [~Potion](#)
  - [Potion, 76](#)
- [~PowerUp](#)
  - [PowerUp, 77](#)

- ~RandomMonster
  - RandomMonster, 86
- ~RoomInstance
  - RoomInstance, 90
- ~ScreenText
  - ScreenText, 100
- ~SearchingMonster
  - SearchingMonster, 101
- ~SlowMonster
  - SlowMonster, 103
- ~SnipingMonster
  - SnipingMonster, 105
- ~SoundEffect
  - SoundEffect, 106
- ~StartingRoom
  - StartingRoom, 108
- ~SwordWeapon
  - SwordWeapon, 109
- ~TreasureRoom
  - TreasureRoom, 110
- ~WallPatrolMonster
  - WallPatrolMonster, 113
- ~Weapon
  - Weapon, 115
- AddCharacter
  - LevelUpSystem, 56
- AddPotion
  - Player, 72
  - RoomInstance, 90
- AddPowerUp
  - Weapon, 116
- AddToCollisionList
  - CollisionSystem, 39
- Animation, 17
  - ~Animation, 18
  - Animation, 17
  - AnimationToSprite, 18
  - Update, 18
- animation.hpp
  - \_ANIMATION\_, 139
- AnimationDeath
  - Animationhandler.hpp, 140
- AnimationDown
  - Animationhandler.hpp, 140
- AnimationHandler, 20
  - ~AnimationHandler, 20
  - AnimationHandler, 20
  - getAnimation, 21
  - setAnimation, 21
- Animationhandler.hpp
  - \_ANIMATIONHANDLER\_CLASS\_, 140
  - AnimationDeath, 140
  - AnimationDown, 140
  - AnimationIdle, 140
  - AnimationIndex, 140
  - AnimationLeft, 140
  - AnimationPS, 140
  - AnimationRight, 140
  - AnimationUp, 140
  - Count, 140
  - animationHandler\_
    - Character, 36
  - AnimationIdle
    - Animationhandler.hpp, 140
  - AnimationIndex
    - Animationhandler.hpp, 140
  - AnimationLeft
    - Animationhandler.hpp, 140
  - AnimationPS
    - Animationhandler.hpp, 140
  - AnimationRight
    - Animationhandler.hpp, 140
  - AnimationToSprite
    - Animation, 18
  - AnimationUp
    - Animationhandler.hpp, 140
- Attack
  - BossMonster, 23
  - Monster, 64
  - Player, 73
  - RandomMonster, 86
  - SearchingMonster, 101
  - SlowMonster, 103
  - SnipingMonster, 105
  - WallPatrolMonster, 113
- attackCooldownLeft
  - Character, 36
  - Weapon, 117
- attackCooldownLength\_
  - Character, 36
  - Weapon, 117
- Bitmasks
  - Collision, 13
- BoostDamageValue
  - Weapon, 116
- BossMonster, 22
  - ~BossMonster, 23
  - Attack, 23
  - BossMonster, 23
  - initVariables, 24
  - Move, 24
- BossMonster.hpp
  - \_BOSS\_MONSTER\_CLASS\_, 126
- BossRoom, 24
  - ~BossRoom, 25
  - BossRoom, 25
  - Enter, 25
  - setTiles, 26
- BossSpawner, 26
  - ~BossSpawner, 27
  - BossSpawner, 26
  - SpawnMonster, 27
- BossSpawner.hpp
  - \_BOSS\_SPAWNER\_CLASS\_, 129
- BoundingBoxTest
  - Collision, 11

BowWeapon, 27  
     ~BowWeapon, 28  
     BowWeapon, 28  
     Use, 28  
 BowWeapon.hpp  
     \_BOWWEAPON\_CLASS\_, 146  
  
 C\_PIXELS  
     game.cpp, 158  
 C\_PIXELS\_delta  
     character.cpp, 123  
 C\_PIXELS\_X  
     character.cpp, 123  
 C\_PIXELS\_Y  
     character.cpp, 123  
 C\_SCALE  
     character.cpp, 124  
 C\_X  
     character.cpp, 124  
 CanAttack  
     Character, 36  
 CanDash  
     Player, 75  
 Character, 28  
     ~Character, 30  
     animationHandler\_, 36  
     attackCooldownLeft, 36  
     attackCooldownLength\_, 36  
     CanAttack, 36  
     Character, 30  
     characterProjectileType\_, 37  
     currentMaxHitpoints\_, 37  
     currentSpeed\_, 37  
     Dead, 30  
     defaultMaxHitpoints\_, 37  
     defaultSpeed\_, 37  
     emptyList, 31  
     Equip, 31  
     generalUpdate, 31  
     GetAttackCooldownLeft, 31  
     GetAttackCooldownLength, 31  
     GetHitPoints, 31  
     GetMaxHP, 32  
     hasAnimation\_, 37  
     HasWeapon, 32  
     Heal, 32  
     hitpoints\_, 37  
     Idle, 32  
     initVariables, 32  
     invincibility\_frame\_, 37  
     IsAlive, 33  
     left\_or\_right\_, 38  
     Move, 33  
     MoveDown, 33  
     MoveLeft, 33  
     MoveRight, 34  
     MoveUp, 34  
     ResetAttackCooldown, 35  
     ResetCharacterToBeAlive, 35  
     RevertMove, 35  
     SetNormalSpeed, 35  
     shotProjectileList, 35  
     startPos, 38  
     TakeDamage, 35  
     Update, 36  
     updateAttackCooldown, 36  
     weapon\_, 38  
 character  
     ICollidable, 51  
 character.cpp  
     C\_PIXELS\_delta, 123  
     C\_PIXELS\_X, 123  
     C\_PIXELS\_Y, 123  
     C\_SCALE, 124  
     C\_X, 124  
 character.hpp  
     \_CHARACTER\_CLASS\_, 124  
 characterLevelMap  
     LevelUpSystem, 57  
 characterProjectileType\_  
     Character, 37  
 CircleTest  
     Collision, 11  
 clampPosToRoom  
     Monster, 64  
 cleared\_  
     RoomInstance, 96  
 ClearInventory  
     Player, 73  
 Collision, 11  
     Bitmasks, 13  
     BoundingBoxTest, 11  
     CircleTest, 11  
     CreateTextureAndBitmask, 12  
     GetSpriteCenter, 12  
     GetSpriteSize, 12  
     PixelPerfectTest, 12  
 Collision::BitmaskManager, 21  
     ~BitmaskManager, 21  
     CreateMask, 21  
     GetMask, 22  
     GetPixel, 22  
 Collision::OrientedBoundingBox, 69  
     OrientedBoundingBox, 70  
     Points, 70  
     ProjectOntoAxis, 70  
 CollisionSystem, 38  
     AddToCollisionList, 39  
     CollisionSystem, 39  
     ProcessCollisionList, 39  
     RemoveFromCollisionList, 39  
 CollisionSystem.hpp  
     \_COLLISIONSYSTEM\_CLASS\_, 161  
 colour\_  
     Potion, 77  
 Connect  
     RoomInstance, 91

- cooldown\_
  - Weapon, [117](#)
- coords\_
  - RoomInstance, [96](#)
- Count
  - Animationhandler.hpp, [140](#)
  - roomInstance.hpp, [152](#)
- CreateDungeon
  - Map, [60](#)
- CreateExit
  - RoomInstance, [91](#)
- CreateMask
  - Collision::BitmaskManager, [21](#)
- CreateSpriteFrom
  - spritehelper, [14](#)
- CreateTextureAndBitmask
  - Collision, [12](#)
- currentDamage\_
  - Weapon, [117](#)
- currentMaxHitpoints\_
  - Character, [37](#)
- currentSpeed\_
  - Character, [37](#)
- damageBoostModifier
  - Weapon, [117](#)
- Dash
  - Player, [73](#)
- Dead
  - Character, [30](#)
- defaultDamage\_
  - Weapon, [117](#)
- defaultMaxHitpoints\_
  - Character, [37](#)
- defaultSpeed\_
  - Character, [37](#)
- deleteMonster
  - RoomInstance, [91](#)
- Direction
  - roomInstance.hpp, [151](#)
- direction, [13](#)
  - GetOppositeDir, [13](#)
- directionsLeft\_
  - RoomInstance, [96](#)
- DirToVec
  - Map, [60](#)
- Down
  - roomInstance.hpp, [152](#)
- emptyList
  - Character, [31](#)
- EnemyProjectile
  - Projectile, [79](#)
- Enter
  - BossRoom, [25](#)
  - RoomInstance, [91](#)
- Entity, [39](#)
  - ~Entity, [41](#)
  - Entity, [40](#), [41](#)
  - GetBaseBoxAt, [41](#)
  - GetOldPosition, [42](#)
  - GetPos, [42](#)
  - GetPosI, [42](#)
  - GetSprite, [42](#)
  - GetSpriteBounds, [43](#)
  - GetSpriteCenter, [43](#)
  - GetSpritePosition, [43](#)
  - initSprite, [43](#)
  - oldPos\_, [44](#)
  - pos\_, [44](#)
  - Render, [43](#)
  - SetPos, [44](#)
  - SetPosAndOldPos, [44](#)
  - sprite\_, [44](#)
  - texture\_, [45](#)
- entity.hpp
  - \_ENTITY\_CLASS\_, [157](#)
- EntityType
  - ICollidable, [51](#)
- Equip
  - Character, [31](#)
- Events
  - Game, [47](#)
- Exit
  - RoomInstance, [92](#)
- FloorTile, [45](#)
  - FloorTile, [45](#)
- FrontWallTile, [46](#)
  - FrontWallTile, [46](#)
- GainXP
  - LevelUpInstance, [54](#)
  - LevelUpSystem, [56](#)
- Game, [46](#)
  - ~Game, [47](#)
  - Events, [47](#)
  - Game, [47](#)
  - RenderGame, [47](#)
  - Running, [47](#)
  - UpdateGame, [48](#)
- game.cpp
  - C\_PIXELS, [158](#)
- game.hpp
  - \_GAME\_CLASS\_, [158](#)
- Gamebar, [48](#)
  - Gamebar, [48](#), [49](#)
  - Render, [49](#)
  - RenderInventory, [49](#)
  - Update, [49](#)
- gamebar.hpp
  - \_GAMEBAR\_CLASS\_, [160](#)
- generalUpdate
  - Character, [31](#)
- getAnimation
  - AnimationHandler, [21](#)
- GetAttackCooldown
  - Weapon, [116](#)

- GetAttackCooldownLeft
  - Character, [31](#)
- GetAttackCooldownLength
  - Character, [31](#)
- GetBaseBoxAt
  - Entity, [41](#)
- GetBoundingBox
  - ICollidable, [51](#)
- getBoundingBox
  - RoomTile, [98](#)
- GetColour
  - Potion, [76](#)
- GetCoords
  - RoomInstance, [92](#)
- GetCurrentRoom
  - Map, [61](#)
- GetDamage
  - Projectile, [79](#)
- GetDashCooldownLeft
  - Player, [73](#)
- GetDashCooldownLength
  - Player, [73](#)
- GetDirection
  - Projectile, [80](#)
- GetDistanceLifeSpan
  - Projectile, [80](#)
- getDistanceToPlayer
  - Monster, [65](#)
- GetEntityType
  - ICollidable, [52](#)
- GetEntranceInDirection
  - RoomInstance, [92](#)
- GetHealthIncrease
  - Potion, [76](#)
- GetHitPoints
  - Character, [31](#)
- GetHPModifier
  - LevelUpInstance, [54](#)
  - LevelUpSystem, [56](#)
- GetInventory
  - Player, [73](#)
- GetLevel
  - LevelUpInstance, [55](#)
  - LevelUpSystem, [57](#)
- GetMask
  - Collision::BitmaskManager, [22](#)
- GetMaxHP
  - Character, [32](#)
- GetMonsterAmount
  - MonsterSpawner, [68](#)
- GetMonsters
  - RoomInstance, [92](#)
- GetOldPosition
  - Entity, [42](#)
- GetOppositeDir
  - direction, [13](#)
- GetPixel
  - Collision::BitmaskManager, [22](#)
- GetPlayer
  - Monster, [65](#)
- GetPos
  - Entity, [42](#)
- GetPosl
  - Entity, [42](#)
- getPosition
  - RoomTile, [98](#)
- GetPotions
  - RoomInstance, [92](#)
- getPowerUpCount
  - Weapon, [116](#)
- GetProjectileSpeed
  - Projectile, [80](#)
- getRandomMonster
  - MonsterSpawner, [68](#)
- GetRange
  - Weapon, [116](#)
- getRefreshRate
  - util::IPlatform, [53](#)
  - util::LinuxPlatform, [58](#)
  - util::WindowsPlatform, [120](#)
- GetRoomInDir
  - Map, [61](#)
  - RoomInstance, [93](#)
- getRoomTilesAt
  - RoomInstance, [93](#)
- getScreenScalingFactor
  - util::IPlatform, [53](#)
  - util::LinuxPlatform, [58](#)
  - util::WindowsPlatform, [120](#)
- getSize
  - RoomTile, [98](#)
- GetSpawnRoom
  - Map, [61](#)
- GetSprite
  - Entity, [42](#)
- getSprite
  - RoomTile, [98](#)
- GetSpriteBounds
  - Entity, [43](#)
- GetSpriteCenter
  - Collision, [12](#)
  - Entity, [43](#)
- GetSpritePosition
  - Entity, [43](#)
- GetSpriteSize
  - Collision, [12](#)
- GetStartPosition
  - Projectile, [80](#)
- getTiles
  - RoomInstance, [93](#)
- GetTimeExisted
  - Projectile, [81](#)
- GetTimeLifeSpan
  - Projectile, [81](#)
- GetType
  - Projectile, [81](#)

- GreenPotion, 50
  - GreenPotion, 50
- hasAnimation\_
  - Character, 37
- HasDirectionsLeft
  - RoomInstance, 93
- HasHit
  - Projectile, 81
- HasWeapon
  - Character, 32
- Heal
  - Character, 32
- healthbar\_
  - Monster, 66
- healthIncrease\_
  - Potion, 77
- HealthPotions.hpp
  - \_HEALTHPOTIONS\_CLASS\_, 142
- Hit
  - Projectile, 82
- hitpoints\_
  - Character, 37
- ICollidable, 50
  - ~ICollidable, 51
  - character, 51
  - EntityType, 51
  - GetBoundingBox, 51
  - GetEntityType, 52
  - ProcessCollision, 52
  - projectile, 51
  - tile, 51
- ICollidable.hpp
  - \_COLLIDABLE\_INTERFACE\_, 162
- Idle
  - Character, 32
- initSprite
  - Entity, 43
- initVariables
  - BossMonster, 24
  - Character, 32
  - Monster, 65
  - Player, 74
  - RandomMonster, 87
  - SearchingMonster, 101
  - SlowMonster, 103
  - SnipingMonster, 105
  - WallPatrolMonster, 113
- inRangeOfPlayer
  - Monster, 65
- invincibility\_frame\_
  - Character, 37
- IsAlive
  - Character, 33
  - Projectile, 82
- IsBossRoomCleared
  - Map, 61
- IsCleared
  - RoomInstance, 93
- IsDashing
  - Player, 75
- isPenetratable
  - RoomTile, 99
- IsVisisted
  - RoomInstance, 94
- isWalkable
  - RoomTile, 99
- Kill
  - Projectile, 82
- Left
  - roomInstance.hpp, 152
- left\_or\_right\_
  - Character, 38
- LevelUp
  - LevelUpInstance, 55
  - LevelUpSystem, 57
- LevelUpInstance, 54
  - GainXP, 54
  - GetHPModifier, 54
  - GetLevel, 55
  - LevelUp, 55
  - LevelUpInstance, 54
- LevelUpInstance.hpp
  - \_LevelUpInstance\_CLASS\_, 170
- LevelUpSystem, 55
  - AddCharacter, 56
  - characterLevelMap, 57
  - GainXP, 56
  - GetHPModifier, 56
  - GetLevel, 57
  - LevelUp, 57
- LevelUpSystem.hpp
  - \_LevelUpSystem\_CLASS\_, 171
- LinuxPlatform
  - util::LinuxPlatform, 58
- llong
  - Types.hpp, 176
- main
  - Main.cpp, 172
- Main.cpp
  - main, 172
- MANIFEST\_RESOURCE\_ID
  - Resource.h, 166
- Map, 59
  - ~Map, 60
  - CreateDungeon, 60
  - DirToVec, 60
  - GetCurrentRoom, 61
  - GetRoomInDir, 61
  - GetSpawnRoom, 61
  - IsBossRoomCleared, 61
  - Map, 60
  - MovePlayer, 62
  - RenderCurrentRoom, 62

- ResetMap, 62
- map.hpp
  - \_MAP\_, 150
- maxPowerUps
  - Weapon, 118
- Monster, 63
  - ~Monster, 64
  - Attack, 64
  - clampPosToRoom, 64
  - getDistanceToPlayer, 65
  - GetPlayer, 65
  - healthbar\_, 66
  - initVariables, 65
  - inRangeOfPlayer, 65
  - Monster, 64
  - Move, 65
  - movedLastTick\_, 67
  - moveTowardsPlayer, 65
  - player\_, 67
  - Render, 66
  - ReturnPotion, 66
  - SetTarget, 66
  - staticDamage\_, 67
  - Update, 66
- monster.hpp
  - \_MONSTER\_CLASS\_, 127
- monsterCleared
  - RoomInstance, 94
- monsterCount\_
  - MonsterSpawner, 69
- monsters\_
  - RoomInstance, 96
- MonsterSP
  - MonsterSpawner.hpp, 130
- MonsterSpawner, 67
  - ~MonsterSpawner, 68
  - GetMonsterAmount, 68
  - getRandomMonster, 68
  - monsterCount\_, 69
  - MonsterSpawner, 68
  - monsterTypeCount\_, 69
  - SetMonsterAmount, 68
  - SpawnMonster, 69
- MonsterSpawner.hpp
  - \_MONSTERSPAWNER\_CLASS\_, 130
  - MonsterSP, 130
  - MonsterSpawnerUP, 130
- MonsterSpawnerUP
  - MonsterSpawner.hpp, 130
- monsterTypeCount\_
  - MonsterSpawner, 69
- Move
  - BossMonster, 24
  - Character, 33
  - Monster, 65
  - RandomMonster, 87
  - SearchingMonster, 102
  - SlowMonster, 104
  - SnipingMonster, 105
  - WallPatrolMonster, 113
- movedLastTick\_
  - Monster, 67
- MoveDown
  - Character, 33
- MoveLeft
  - Character, 33
- MovePlayer
  - Map, 62
- MoveRight
  - Character, 34
- moveTowardsPlayer
  - Monster, 65
- MoveUp
  - Character, 34
- NDEBUG
  - PCH.hpp, 163
- oldPos\_
  - Entity, 44
- OrientedBoundingBox
  - Collision::OrientedBoundingBox, 70
- PCH.hpp
  - NDEBUG, 163
  - UNUSED, 163
- Penetrates
  - Projectile, 82
- penetrates\_
  - Weapon, 118
- PI
  - SpriteHelper.cpp, 175
- PixelPerfectTest
  - Collision, 12
- Player, 70
  - ~Player, 71
  - AddPotion, 72
  - Attack, 73
  - CanDash, 75
  - ClearInventory, 73
  - Dash, 73
  - GetDashCooldownLeft, 73
  - GetDashCooldownLength, 73
  - GetInventory, 73
  - initVariables, 74
  - IsDashing, 75
  - Player, 71
  - ResetDashCooldown, 74
  - Update, 74
  - UsePotion, 74
- player.hpp
  - \_PLAYER\_CLASS\_, 137
  - PlayerPS, 137
- player\_
  - Monster, 67
- PlayerProjectile
  - Projectile, 79

- PlayerPS
  - player.hpp, 137
- PlaySound
  - SoundEffect, 106
- Points
  - Collision::OrientedBoundingBox, 70
- pos\_
  - Entity, 44
- positionIsPenetratable
  - RoomInstance, 94
- positionIsWalkable
  - RoomInstance, 94
- Potion, 75
  - ~Potion, 76
  - colour\_, 77
  - GetColour, 76
  - GetHealthIncrease, 76
  - healthIncrease\_, 77
  - Potion, 76
- Potion.hpp
  - \_POTION\_CLASS\_, 143
- potions\_
  - RoomInstance, 96
- PowerUp, 77
  - ~PowerUp, 77
  - PowerUp, 77
- PowerUp.hpp
  - \_POWERUP\_CLASS\_, 144
- powerUps\_
  - Weapon, 118
- ProcessCollision
  - ICollidable, 52
- ProcessCollisionList
  - CollisionSystem, 39
- Projectile, 78
  - EnemyProjectile, 79
  - GetDamage, 79
  - GetDirection, 80
  - GetDistanceLifeSpan, 80
  - GetProjectileSpeed, 80
  - GetStartPosition, 80
  - GetTimeExisted, 81
  - GetTimeLifeSpan, 81
  - GetType, 81
  - HasHit, 81
  - Hit, 82
  - IsAlive, 82
  - Kill, 82
  - Penetrates, 82
  - PlayerProjectile, 79
  - Projectile, 79
  - SetDamage, 83
  - SetDirection, 83
  - SetDistanceLifeSpan, 83
  - SetProjectileSpeed, 84
  - SetSprite, 84
  - SetTimeLifeSpan, 84
  - SetType, 84
  - Type, 79
  - Update, 85
- projectile
  - ICollidable, 51
- Projectile.hpp
  - \_Projectile\_CLASS\_, 145
  - ProjectileUP, 145
- projectileSize\_
  - Weapon, 118
- projectileSpeed\_
  - Weapon, 118
- ProjectileUP
  - Projectile.hpp, 145
- ProjectOntoAxis
  - Collision::OrientedBoundingBox, 70
- RandomFloatBetween
  - randomhelper, 13
- randomhelper, 13
  - RandomFloatBetween, 13
  - RandomIntBetween, 13
- RandomHelper.hpp
  - \_RANDOMHELPER\_, 173
- RandomIntBetween
  - randomhelper, 13
- RandomMonster, 85
  - ~RandomMonster, 86
  - Attack, 86
  - initVariables, 87
  - Move, 87
  - RandomMonster, 86
- RandomMonster.hpp
  - \_RANDOM\_MONSTER\_CLASS\_, 132
- range\_
  - Weapon, 118
- RedPotion, 87
  - RedPotion, 88
- RemoveDirection
  - RoomInstance, 94
- RemoveFromCollisionList
  - CollisionSystem, 39
- RemoveRandomDirection
  - RoomInstance, 95
- Render
  - Entity, 43
  - Gamebar, 49
  - Monster, 66
  - RoomInstance, 95
- RenderCurrentRoom
  - Map, 62
- RenderGame
  - Game, 47
- RenderInventory
  - Gamebar, 49
- renderSpriteBackground
  - RoomInstance, 95
- ResetAttackCooldown
  - Character, 35
- ResetCharacterToBeAlive



- Character, [35](#)
- ResetDashCooldown
  - Player, [74](#)
- ResetMap
  - Map, [62](#)
- Resource.h
  - MANIFEST\_RESOURCE\_ID, [166](#)
  - WIN32\_ICON\_MAIN, [166](#)
- ReturnPotion
  - Monster, [66](#)
- RevertMove
  - Character, [35](#)
- Right
  - roomInstance.hpp, [152](#)
- roomBackground
  - RoomInstance, [96](#)
- RoomInstance, [88](#)
  - ~RoomInstance, [90](#)
  - AddPotion, [90](#)
  - cleared\_, [96](#)
  - Connect, [91](#)
  - coords\_, [96](#)
  - CreateExit, [91](#)
  - deleteMonster, [91](#)
  - directionsLeft\_, [96](#)
  - Enter, [91](#)
  - Exit, [92](#)
  - GetCoords, [92](#)
  - GetEntranceInDirection, [92](#)
  - GetMonsters, [92](#)
  - GetPotions, [92](#)
  - GetRoomInDir, [93](#)
  - getRoomTilesAt, [93](#)
  - getTiles, [93](#)
  - HasDirectionsLeft, [93](#)
  - IsCleared, [93](#)
  - IsVisisted, [94](#)
  - monsterCleared, [94](#)
  - monsters\_, [96](#)
  - positionIsPenetratable, [94](#)
  - positionIsWalkable, [94](#)
  - potions\_, [96](#)
  - RemoveDirection, [94](#)
  - RemoveRandomDirection, [95](#)
  - Render, [95](#)
  - renderSpriteBackground, [95](#)
  - roomBackground, [96](#)
  - RoomInstance, [90](#)
  - roomSize\_, [96](#)
  - roomTexture, [96](#)
  - setTiles, [95](#)
  - spawner\_, [97](#)
  - tileVector\_, [97](#)
  - visited\_, [97](#)
- roomInstance.hpp
  - Count, [152](#)
  - Direction, [151](#)
  - Down, [152](#)
  - Left, [152](#)
  - Right, [152](#)
  - Up, [152](#)
- roomSize\_
  - RoomInstance, [96](#)
- roomTexture
  - RoomInstance, [96](#)
- RoomTile, [97](#)
  - getBoundingBox, [98](#)
  - getPosition, [98](#)
  - getSize, [98](#)
  - getSprite, [98](#)
  - isPenetratable, [99](#)
  - isWalkable, [99](#)
  - RoomTile, [98](#)
- RotateSprite
  - spritehelper, [14](#)
- Running
  - Game, [47](#)
- ScreenText, [99](#)
  - ~ScreenText, [100](#)
  - ScreenText, [99](#)
- SearchingMonster, [100](#)
  - ~SearchingMonster, [101](#)
  - Attack, [101](#)
  - initVariables, [101](#)
  - Move, [102](#)
  - SearchingMonster, [101](#)
- SearchingMonster.hpp
  - \_SEARCHING\_MONSTER\_CLASS\_, [133](#)
- setAnimation
  - AnimationHandler, [21](#)
- SetDamage
  - Projectile, [83](#)
- SetDirection
  - Projectile, [83](#)
- SetDistanceLifeSpan
  - Projectile, [83](#)
- setIcon
  - util::IPlatform, [53](#)
  - util::LinuxPlatform, [58](#)
  - util::WindowsPlatform, [120](#)
- SetMonsterAmount
  - MonsterSpawner, [68](#)
- SetNormalSpeed
  - Character, [35](#)
- SetOriginBottomCenter
  - spritehelper, [14](#)
- SetPos
  - Entity, [44](#)
- SetPosAndOldPos
  - Entity, [44](#)
- SetProjectileSpeed
  - Projectile, [84](#)
- SetScale
  - spritehelper, [14](#)
- SetSprite
  - Projectile, [84](#)

- SetTarget
  - Monster, 66
- SetTextureRect
  - Weapon, 116
- setTiles
  - BossRoom, 26
  - RoomInstance, 95
  - StartingRoom, 108
  - TreasureRoom, 110
- SetTimeLifeSpan
  - Projectile, 84
- SetType
  - Projectile, 84
- shotProjectileList
  - Character, 35
- SlowMonster, 102
  - ~SlowMonster, 103
  - Attack, 103
  - initVariables, 103
  - Move, 104
  - SlowMonster, 103
- SlowMonster.hpp
  - \_SLOW\_MONSTER\_CLASS\_, 134
- SnipingMonster, 104
  - ~SnipingMonster, 105
  - Attack, 105
  - initVariables, 105
  - Move, 105
  - SnipingMonster, 105
- SnipingMonster.hpp
  - \_SNIPING\_MONSTER\_CLASS\_, 135
- SoundEffect, 106
  - ~SoundEffect, 106
  - PlaySound, 106
  - SoundEffect, 106
- spawner\_
  - RoomInstance, 97
- SpawnMonster
  - BossSpawner, 27
  - MonsterSpawner, 69
- speed\_
  - Weapon, 118
- sprite\_
  - Entity, 44
  - Weapon, 118
- spritehelper, 14
  - CreateSpriteFrom, 14
  - RotateSprite, 14
  - SetOriginBottomCenter, 14
  - SetScale, 14
- SpriteHelper.cpp
  - PI, 175
- SpriteHelper.hpp
  - \_SPRITEHELPER\_, 176
- src/Actors/character.cpp, 123
- src/Actors/character.hpp, 124, 125
- src/Actors/Monsters/BossMonster.cpp, 126
- src/Actors/Monsters/BossMonster.hpp, 126
- src/Actors/Monsters/monster.cpp, 127
- src/Actors/Monsters/monster.hpp, 127, 128
- src/Actors/Monsters/MonsterSpawner/BossSpawner.cpp, 128
- src/Actors/Monsters/MonsterSpawner/BossSpawner.hpp, 128, 129
- src/Actors/Monsters/MonsterSpawner/MonsterSpawner.cpp, 129
- src/Actors/Monsters/MonsterSpawner/MonsterSpawner.hpp, 129, 131
- src/Actors/Monsters/RandomMonster.cpp, 131
- src/Actors/Monsters/RandomMonster.hpp, 131, 132
- src/Actors/Monsters/SearchingMonster.cpp, 132
- src/Actors/Monsters/SearchingMonster.hpp, 132, 133
- src/Actors/Monsters/SlowMonster.cpp, 133
- src/Actors/Monsters/SlowMonster.hpp, 133, 134
- src/Actors/Monsters/SnipingMonster.cpp, 134
- src/Actors/Monsters/SnipingMonster.hpp, 135
- src/Actors/Monsters/WallPatrolMonster.cpp, 136
- src/Actors/Monsters/WallPatrolMonster.hpp, 136
- src/Actors/player.cpp, 137
- src/Actors/player.hpp, 137, 138
- src/Animation/animation.cpp, 138
- src/Animation/animation.hpp, 138, 139
- src/Animation/Animationhandler.cpp, 139
- src/Animation/Animationhandler.hpp, 139, 141
- src/Combat/CircularProjectile.hpp, 141
- src/Combat/Health/HealthPotions.hpp, 141, 142
- src/Combat/Health/Potion.cpp, 142
- src/Combat/Health/Potion.hpp, 143
- src/Combat/PowerUps/PowerUp.hpp, 143, 144
- src/Combat/projectile.cpp, 144
- src/Combat/Projectile.hpp, 144, 145
- src/Combat/Weapons/BowWeapon.cpp, 146
- src/Combat/Weapons/BowWeapon.hpp, 146, 147
- src/Combat/Weapons/SwordWeapon.cpp, 147
- src/Combat/Weapons/SwordWeapon.hpp, 147, 148
- src/Combat/Weapons/Weapon.cpp, 148
- src/Combat/Weapons/Weapon.hpp, 148, 149
- src/Dungeon/map.cpp, 149
- src/Dungeon/map.hpp, 149, 150
- src/Dungeon/roomInstance.cpp, 151
- src/Dungeon/roomInstance.hpp, 151, 152
- src/Dungeon/specialrooms/BossRoom.cpp, 153
- src/Dungeon/specialrooms/BossRoom.hpp, 153
- src/Dungeon/specialrooms/startingRoom.cpp, 154
- src/Dungeon/specialrooms/startingRoom.hpp, 154
- src/Dungeon/specialrooms/TreasureRoom.cpp, 154
- src/Dungeon/specialrooms/TreasureRoom.hpp, 155
- src/Dungeon/Tiles/roomTile.cpp, 155
- src/Dungeon/Tiles/roomTile.hpp, 155, 156
- src/entity.cpp, 156
- src/entity.hpp, 156, 157
- src/game.cpp, 157
- src/game.hpp, 158, 159
- src/gamebar.cpp, 159
- src/gamebar.hpp, 160
- src/Interfaces/CollisionSystem.hpp, 161

- src/Interfaces/ICollidable.hpp, 162
- src/PCH.hpp, 163, 164
- src/Platform/IPlatform.hpp, 164, 165
- src/Platform/Platform.hpp, 165
- src/Platform/Unix/LinuxPlatform.cpp, 165
- src/Platform/Unix/LinuxPlatform.hpp, 165, 166
- src/Platform/Win32/Resource.h, 166, 167
- src/Platform/Win32/WindowsPlatform.cpp, 167
- src/Platform/Win32/WindowsPlatform.hpp, 167
- src/readme.md, 168
- src/Utility/Collision.cpp, 168
- src/Utility/Collision.hpp, 168, 169
- src/Utility/FileSystem.hpp, 169
- src/Utility/LevelUpInstance.cpp, 170
- src/Utility/LevelUpInstance.hpp, 170, 171
- src/Utility/LevelUpSystem.cpp, 171
- src/Utility/LevelUpSystem.hpp, 171, 172
- src/Utility/Main.cpp, 172
- src/Utility/RandomHelper.cpp, 172
- src/Utility/RandomHelper.hpp, 173
- src/Utility/ScreenText.cpp, 173
- src/Utility/ScreenText.hpp, 174
- src/Utility/Sounds/SoundEffects.cpp, 174
- src/Utility/Sounds/SoundEffects.hpp, 174
- src/Utility/SpriteHelper.cpp, 175
- src/Utility/SpriteHelper.hpp, 175, 176
- src/Utility/Types.hpp, 176, 177
- StartingRoom, 107
  - ~StartingRoom, 108
  - setTiles, 108
  - StartingRoom, 107, 108
- startingRoom.hpp
  - \_STARTING\_ROOM\_CLASS\_, 154
- startPos
  - Character, 38
- staticDamage\_
  - Monster, 67
- SwordWeapon, 108
  - ~SwordWeapon, 109
  - SwordWeapon, 109
  - Use, 109
- SwordWeapon.hpp
  - \_SWORDWEAPON\_CLASS\_, 147
- TakeDamage
  - Character, 35
- texture\_
  - Entity, 45
  - Weapon, 119
- tile
  - ICollidable, 51
- tileVector\_
  - RoomInstance, 97
- toggleFullscreen
  - util::IPlatform, 53
  - util::LinuxPlatform, 59
  - util::WindowsPlatform, 120
- TreasureRoom, 109
  - ~TreasureRoom, 110
  - setTiles, 110
  - TreasureRoom, 110
- Type
  - Projectile, 79
- Types.hpp
  - llong, 176
  - uchar, 176
  - uint, 177
  - ullong, 177
  - ushort, 177
- uchar
  - Types.hpp, 176
- uint
  - Types.hpp, 177
- ullong
  - Types.hpp, 177
- UnBoostDamageValue
  - Weapon, 116
- UNUSED
  - PCH.hpp, 163
- Up
  - roomInstance.hpp, 152
- Update
  - Animation, 18
  - Character, 36
  - Gamebar, 49
  - Monster, 66
  - Player, 74
  - Projectile, 85
- updateAttackCooldown
  - Character, 36
- UpdateGame
  - Game, 48
- Use
  - BowWeapon, 28
  - SwordWeapon, 109
  - Weapon, 117
- UsePotion
  - Player, 74
- ushort
  - Types.hpp, 177
- util, 15
- util::IPlatform, 52
  - ~IPlatform, 52
  - getRefreshRate, 53
  - getScreenScalingFactor, 53
  - setIcon, 53
  - toggleFullscreen, 53
- util::LinuxPlatform, 58
  - getRefreshRate, 58
  - getScreenScalingFactor, 58
  - LinuxPlatform, 58
  - setIcon, 58
  - toggleFullscreen, 59
- util::WindowsPlatform, 119
  - getRefreshRate, 120
  - getScreenScalingFactor, 120
  - setIcon, 120

- [toggleFullscreen](#), [120](#)
  - [WindowsPlatform](#), [119](#)
- [VioletPotion](#), [111](#)
  - [VioletPotion](#), [111](#)
- [visited\\_](#)
  - [RoomInstance](#), [97](#)
- [WallPatrolMonster](#), [112](#)
  - [~WallPatrolMonster](#), [113](#)
  - [Attack](#), [113](#)
  - [initVariables](#), [113](#)
  - [Move](#), [113](#)
  - [WallPatrolMonster](#), [112](#)
- [WallPatrolMonster.hpp](#)
  - [\\_WALL\\_PATROL\\_MONSTER\\_CLASS\\_](#), [136](#)
- [WallTile](#), [114](#)
  - [WallTile](#), [114](#)
- [Weapon](#), [114](#)
  - [~Weapon](#), [115](#)
  - [AddPowerUp](#), [116](#)
  - [attackCooldownLeft](#), [117](#)
  - [attackCooldownLength\\_](#), [117](#)
  - [BoostDamageValue](#), [116](#)
  - [cooldown\\_](#), [117](#)
  - [currentDamage\\_](#), [117](#)
  - [damageBoostModifier](#), [117](#)
  - [defaultDamage\\_](#), [117](#)
  - [GetAttackCooldown](#), [116](#)
  - [getPowerUpCount](#), [116](#)
  - [GetRange](#), [116](#)
  - [maxPowerUps](#), [118](#)
  - [penetrates\\_](#), [118](#)
  - [powerUps\\_](#), [118](#)
  - [projectileSize\\_](#), [118](#)
  - [projectileSpeed\\_](#), [118](#)
  - [range\\_](#), [118](#)
  - [SetTextureRect](#), [116](#)
  - [speed\\_](#), [118](#)
  - [sprite\\_](#), [118](#)
  - [texture\\_](#), [119](#)
  - [UnBoostDamageValue](#), [116](#)
  - [Use](#), [117](#)
  - [Weapon](#), [115](#)
- [Weapon.hpp](#)
  - [\\_WEAPON\\_CLASS\\_](#), [148](#)
- [weapon\\_](#)
  - [Character](#), [38](#)
- [WIN32\\_ICON\\_MAIN](#)
  - [Resource.h](#), [166](#)
- [WindowsPlatform](#)
  - [util::WindowsPlatform](#), [119](#)
- [YellowPotion](#), [121](#)
  - [YellowPotion](#), [121](#)